NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

AUTOMATED CREATION OF LABELED POINTCLOUD DATASETS IN SUPPORT OF MACHINE LEARNING–BASED PERCEPTION

by

Andrew K. Watson

December 2017

Thesis Advisor:	Douglas Horner
Co-Advisor:	Mathias Kölsch
Second Reader:	Michael McCarrin

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2017	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE AUTOMATED CREATION OF LABELED POINTCLOUD DATASETS IN SUPPORT OF MACHINE LEARNING–BASED PERCEPTION	**5. FUNDING NUMBERS**
6. AUTHOR(S) Andrew K. Watson	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB number ____N/A ____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Autonomous vehicles continue to struggle with understanding their environments and robotic perception remains an active area of research. Machine learning–based approaches to computer vision, particularly the increasing application of deep neural networks, have been responsible for many of the breakthroughs in robotic perception over the last decade. We propose a three-phase model for improving pointcloud classification. Progress in applying machine learning–based perception to new problem sets is hampered by the difficulty in creating new training data. As such, our primary contribution is a technique to automate the creation of training data for 3D pointcloud classification problems. Our proposed implementation collects synchronized 2D camera images and 3D LIDAR pointclouds, depth clusters each LIDAR frame to spatially segment a scene, correlates each resultant pointcloud segment to a cropped 2D image, and processes each crop through a 2D image classifier to assign a segment label. Our automated implementation produced labeled 3D pointclouds from raw LIDAR collection and, during testing, yielded a small dataset with 81% accuracy of annotations. We also propose a method of scene "context discovery" to boost pointcloud classification performance. Our approach explores a method to scrape regionally geotagged media for processing through an object-detection neural network. We develop a database mapping of object-type spatial relationships in a specific physical environment and propose applying these relationships as weights to boost pointcloud classifier performance.

14. SUBJECT TERMS neural network, dataset annotation, LIDAR, computer vision, scene context	**15. NUMBER OF PAGES** 103
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

AUTOMATED CREATION OF LABELED POINTCLOUD DATASETS IN SUPPORT OF MACHINE LEARNING–BASED PERCEPTION

Andrew K. Watson
Civilian, United States Government
B.S., University of Illinois, 2003

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2017

Approved by: Douglas Horner, Ph.D.
Thesis Advisor

Mathias Kölsch, Ph.D.
Co-Advisor

Michael McCarrin
Second Reader

Peter Denning, Ph.D.
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Autonomous vehicles continue to struggle with understanding their environments and robotic perception remains an active area of research. Machine learning–based approaches to computer vision, particularly the increasing application of deep neural networks, have been responsible for many of the breakthroughs in robotic perception over the last decade. We propose a three-phase model for improving pointcloud classification. Progress in applying machine learning–based perception to new problem sets is hampered by the difficulty in creating new training data. As such, our primary contribution is a technique to automate the creation of training data for 3D pointcloud classification problems. Our proposed implementation collects synchronized 2D camera images and 3D LIDAR pointclouds, depth clusters each LIDAR frame to spatially segment a scene, correlates each resultant pointcloud segment to a cropped 2D image, and processes each crop through a 2D image classifier to assign a segment label. Our automated implementation produced labeled 3D pointclouds from raw LIDAR collection and, during testing, yielded a small dataset with 81% accuracy of annotations. We also propose a method of scene “context discovery” to boost pointcloud classification performance. Our approach explores a method to scrape regionally geotagged media for processing through an object-detection neural network. We develop a database mapping of object-type spatial relationships in a specific physical environment and propose applying these relationships as weights to boost pointcloud classifier performance.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

2D	Two Dimensional
3D	Three Dimensional
4K	Video Format with Four Times the Resolution of High Definition
COCO	Common Objects in Context
DARPA	Defense Advanced Research Projects Agency
ILSRVC	ImageNet Large Scale Recognition Visual Competition
Inception-v3	3rd Release of the Inception neural network model
LIDAR	Light Detection and Ranging
MJPEG	Motion Joint Photographic Experts Group
MNIST	Modified National Institute of Standards and Technology
PCL	Point Cloud Library
PLA	Polylactic Acid
RGB	Red, Green, and Blue
RGB-D	Red, Green, Blue, and Depth
ROS	Robot Operating System
RViz	ROS Visualization
SSD	Solid State Drive
UDP	User Datagram Protocol
USB	Universal Serial Bus
Voxel	Volumetric Pixel
VOC	Visual Objects Classes

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my wife, Jenna, for all her hard work in helping me complete this research. Without her assistance, it would not have been possible to dedicate the hours required to finish and I am very grateful.

I would also like to thank my advisors, Doug, Mathias, and Michael, for their combined effort over the last year and during the final weeks of processing. Your guidance was top notch and I appreciate you encouraging me to finish despite the difficult time constraints.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Rapid improvements in neural networks over the previous decade promise a range of new capabilities for autonomous systems, particularly in the realm of autonomous vehicles. However, the underlying technology of these improvements, namely deep neural networks, can require a tremendous amount of labeled training data to produce accurate classifiers. Labeled training data typically consists of a collection of dataset records, with each record having a discrete ground-truth label assigned to it. As an example of the scope of such datasets, the most well-known image classification challenge, the ImageNet Large Scale Visualization Recognition Challenge (ILSRVC), provides over 1.2 million hand-annotated images as labeled training data for training image classifiers [1]. This reliance on large, labeled training datasets has the potential to throttle the development of autonomous capabilities for new problem sets. Our research proposes a three phase model for improved pointcloud classification and provides an evaluation of the first phase. Specifically, we evaluate whether well-documented improvements in 2D image classification can produce labeled training data for other realms of robotic perception. We evaluate our proposal on 3D LIDAR data, a type of data which is especially relevant in the field of autonomous vehicles, and leverage an industry-standard neural network to create training data for another neural network. Specifically, we seek to measure the accuracy of annotating 3D LIDAR pointcloud segments with the output of a 2D image classifier. We also conduct an overview of established hand annotation practices and compare hand annotation to our proposed approach.

The application of LIDAR and other pointcloud producing sensors in the field of robotics has historically focused on obstacle avoidance and path planning. For example, LIDAR sensors provided Stanford's "Stanley" vehicle with superior path planning during the 2005 DARPA Grand Challenges [2], and provide similar capabilities to a wide range of autonomous passenger vehicles being road-tested in 2017 [3]. However, obstacle avoidance, a navigation technique, provides an autonomous vehicle with a very limited understanding of its environment – primarily whether a path is navigable or not. A more

sophisticated understanding of an environment, such as the identification of the specific type of obstacle in an autonomous vehicle's path (i.e., a human obstacle versus a trash bag obstacle), can allow for better decisions in response to the nature of an obstacle. A key step towards that end is object recognition and our research aims to bolster creation of the training data required to build the requisite semantic neural network classifiers.

A. PROPOSED APPROACH

We implement and evaluate the first phase of our proposed point classification model. Specifically, we implement an automated software pipeline designed to create labeled training datasets. Notably, the pipeline requires specialized, tandem collection of raw 2D and 3D data, and subsequently processes 2D image crops through an image classifier to create labels for LIDAR pointcloud segments. This is in contrast to the established practice of manual labeling by humans, known as hand annotation. Hand annotation of datasets can be resource intensive (see section 2.B) and LIDAR data's low resolution representation of objects provides significantly less context for a human annotator when determining how to label an object compared to high definition 2D images. As our proposed pipeline creates labels from the output of an industry-standard 2D image classifier, the intent is to leverage existing and future improvements in the image classification on an entirely modular basis. Specifically, our pipeline's image classifier can be replaced with a more accurate classifier, once available, and any such image classification improvements will have an immediate, positive benefit on our pipeline's dataset annotation performance. We note the tremendous improvements in 2D neural network-based image classification in Figure 1, but highlight that Top-1 accuracy, a term used to describe a classifier's top guess for an image's contents, reaches only 81.32% on the ILSVRC2012 benchmark [4], leaving significant room for future improvement. Our pipeline's accuracy will be constrained by the Top-1 accuracy performance of available 2D classifiers.

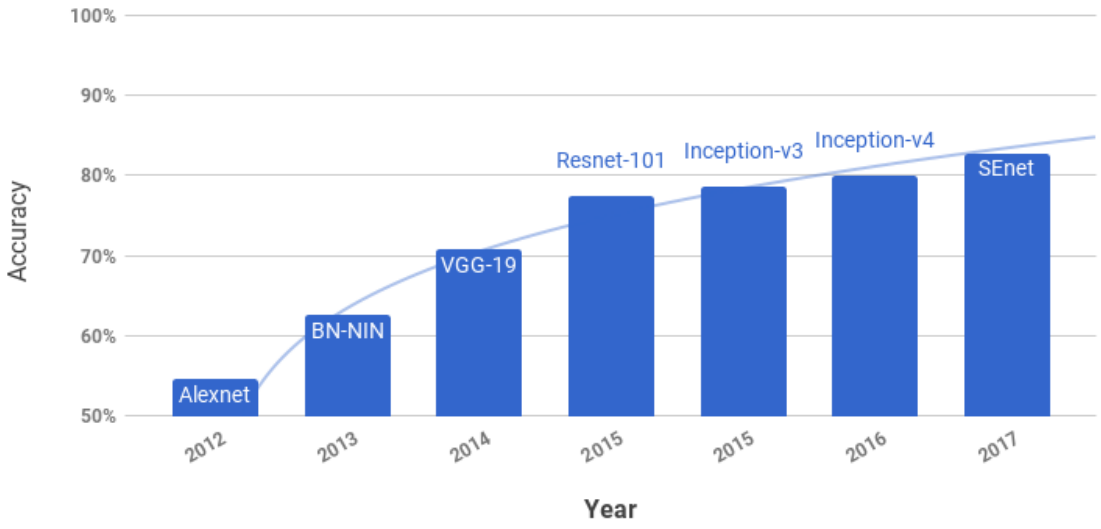


Figure 1. Year Over Year Improvements in Top-1 Image Classification Accuracy on the ILSVRC2012 Validation Set. Adapted from [4], [5], [6].

Our research focuses on a pipeline to create labeled LIDAR data; however, we propose that the technique employed could be broadened to automate the labeling of datasets from other pointcloud-producing sensors as well.

Our research also explores the development of a "context database" describing spatial relationships between object types in a specific physical environment. We discuss a methodology to incorporate the strength of object type relationships into our classification results as a means of increasing accuracy.

B. CONTRIBUTIONS

The primary contribution is the presentation of three phase model for automated 3D pointcloud labeling, scene context discovery, and novel adjustments to established pointcloud classifiers. We accomplish an initial demonstration of automated dataset creation by collecting synchronized camera and pointcloud data and assigning pointcloud labels from the output of an image classifier. The benefit of our approach is the reduction of resource-intensive human annotation from the pointcloud labeling process through the execution of a framework for autonomously labeling large pointcloud datasets. The initial limitations of our fielded demonstration are diminished annotation accuracy, compared to

human annotation, and the discarding of low-confidence segments by our pipeline. We assess our work as potentially useful for producing an initial “first draft” of labels for a human annotator to validate, as opposed to manually selecting a class label from a very large list. Given difficulty in manually labeling pointclouds, we assess that leveraging our pipeline’s output label as a recommendation for a human annotator would prove useful in hastening the pointcloud annotation process. We also provide an exploratory implementation for automating what we characterize as scene *context discovery*. Context discovery, as our model defines it, requires acquiring geotagged media recorded in a desired operating environment and processing the media with object-detection neural networks. We map out spatial relationships between object types in an environment using these results and populate a database. Lastly, we outline two schemes to apply our scene context data to pointcloud classification results, with the intent of improving pointcloud classifier performance.

II. MODEL OVERVIEW

We propose a comprehensive model for classification of pointclouds and we present the model's three phases. We first outline our model's two pre-processing phases—dataset creation and context discovery—and then describe the model's third phase, pointcloud classification. We begin our model overview with a detailed discussion of our automated dataset creation proposal, which we characterize as Phase 1. Phase 1's automated dataset creation methodology begins with the collection of synchronized LIDAR and RGB camera data and executes an automated dataset labeling algorithm. Specifically, our Phase 1 labeling methodology leverages a 2D image classifier, coupled with the RGB camera frames, to create labels for segmented pointclouds from our synchronized data collection. Phase 2 of our model describes "context discovery," a technique we propose to improve classification by producing a database which is keyed to a physical operating environment and contains the assessed likelihood of spatial object pairings for that environment. These pre-calculated likelihoods are applied as weights to Phase 3's classification results, boosting results that are prevalent in the operating environment. Phase 2's context discovery algorithm builds the relational database by processing geotagged media from a desired operating environment.

Following completion of the Phase 1 and Phase 2's pre-processing steps, we further propose a method for pointcloud classification, Phase 3. Our outline of Phase 3 offers three approaches to boosting classifier performance: temporal weighting, absolute weighting, and relational weighting. Temporal weighting leverages previous-frame segment classifications to adjust weights of current-frame segment classifications. Absolute weighting applies the absolute hit-counts from the context discovery database as weights to the pointcloud classifier outputs. Relational weighting does intra-frame analysis and applies a weight to prospective classifications in a given scene based on the prevalence of object relationships in the context discovery database. While our research primarily focuses on evaluating the first pre-processing component of this model, the Phase 1 automated dataset creation, we outline all three phases of our model to provide context and as potential future work. Figure 2 provides a visual overview of the three

phases in our model. We first discuss the two pre-processing phases, dataset creation and context discovery, and finish with a discussion of the Phase 3 pointcloud classifier.

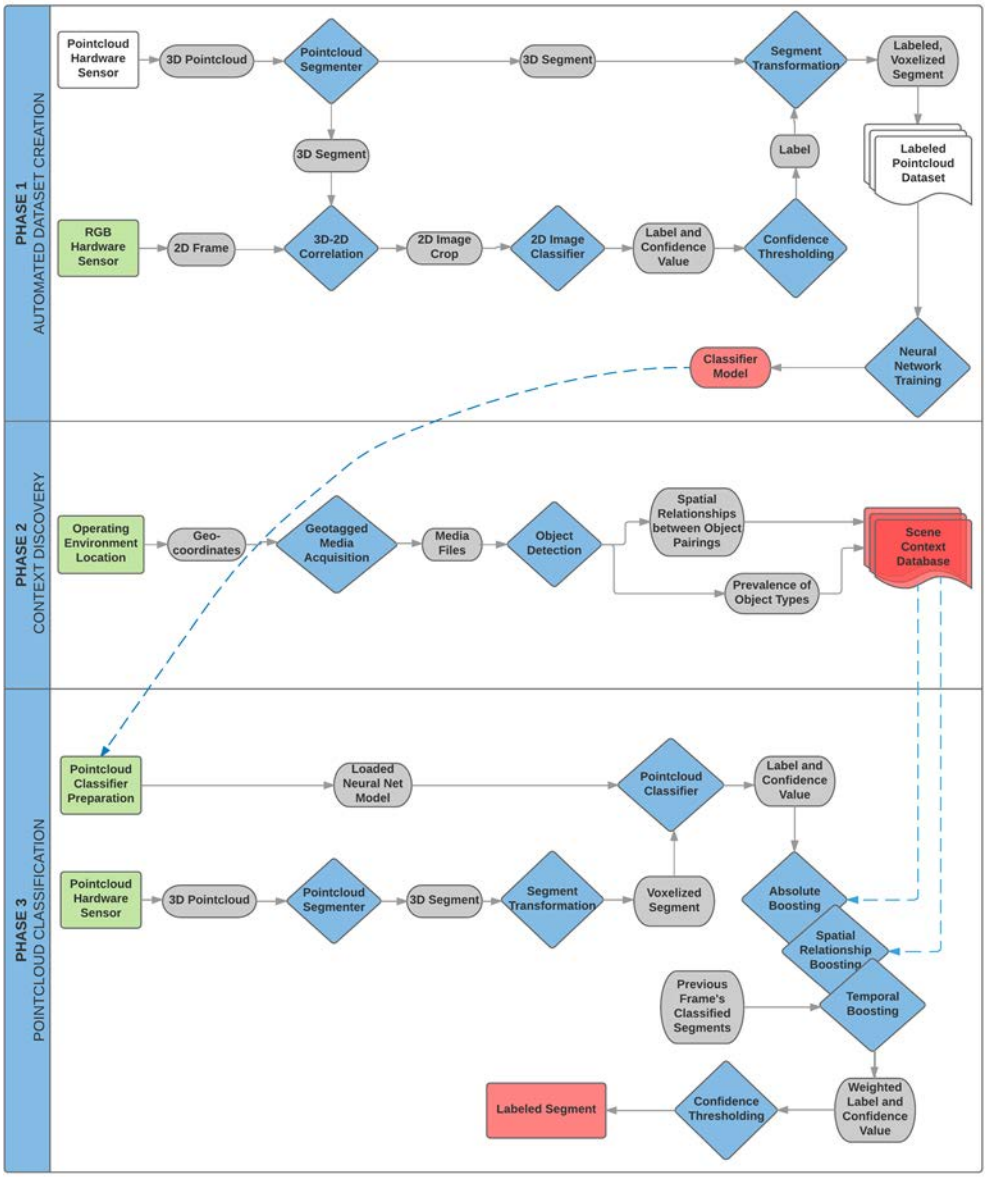


Figure 2. Flowchart of Three-Phase Pointcloud Classification Model

A. PHASE 1: AUTOMATED DATASET CREATION AND TRAINING

Phase 1, automated dataset creation, is the first pre-processing step of our model. We provide an outline for our automated dataset creation methodology and describe the multi-step procedure for automating the labeling of pointcloud segments from raw, synchronized 2D and 3D collection. Figure 3 provides an overview of Phase 1.

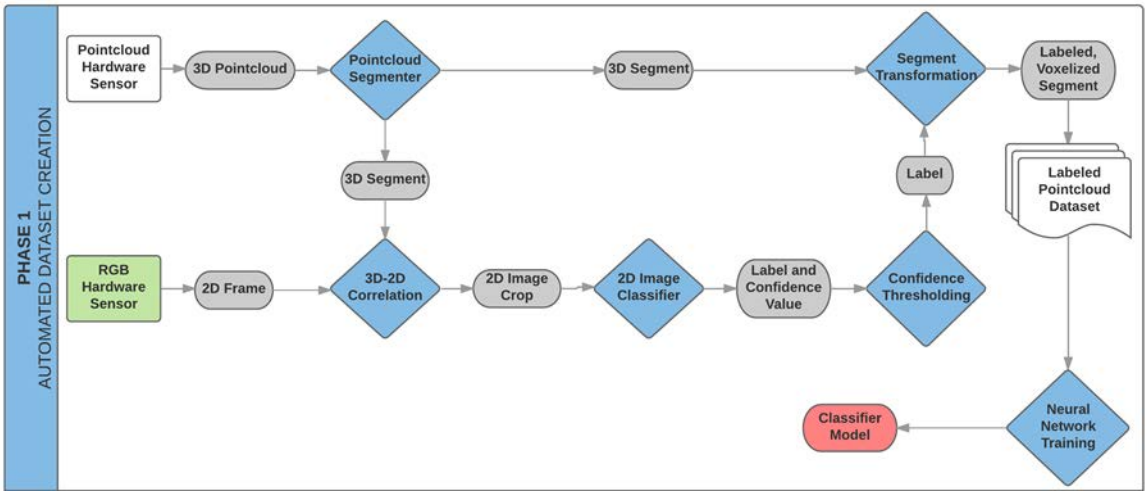


Figure 3. Flowchart of Phase 1, Automated Dataset Creation

1. Synchronized Collection of Pointcloud and RGB Data

The automated dataset creation process we propose first requires the collection and saving of synchronized 2D camera and 3D pointcloud data to an archive. The raw RGB data collected during Phase 1 is leveraged during subsequent Phase 1 steps to derive plain English labels. As such, we propose employing RGB sensors that provide a maximum horizontal field of view during the synchronized collection step. Leveraging hardware featuring a large horizontal field of view will increase the likelihood of collecting multiple look angles at individual objects and could ultimately produce a more comprehensive labeled dataset at the completion of Phase 1. Of note, the requirement for 2D RGB camera data limits Phase 1's data collection to well-lit, daytime operating conditions.

2. Pointcloud Segmentation

The pointcloud segmentation step executes on the pointcloud portion of the synchronized data previously collected in Phase 1. The pointcloud data frames contain a raw, unprocessed representation of the entire scene imaged by a 3D sensor. Recall that Phase 1's aim is to produce a dataset of labeled, distinct objects from this raw collection and, as such, pointcloud segmentation is applied to each pointcloud frame. The segmentation process divides the pointcloud scene into individual clusters of points, with each cluster or segment representing a distinct object in the scene. These segments remain of unknown object type and subsequent steps aim to identify and label each segment.

3. 3D-2D Correlation

The previous step produced pointcloud segments for discrete objects in a pointcloud scene. We now begin a multi-step process of determining a label for a segment, beginning with mapping a 3D pointcloud segment to a 2D image. Recall that Phase 1 required synchronized 2D images to be collected alongside 3D pointcloud data. We begin the 3D-2D correlation step by loading the synchronized 2D image that was collected at the same moment as the pointcloud segment we seek to label. Unlike the now-segmented pointcloud data, the 2D frame still represents the entire scene imaged by the RGB sensor at the time of collection. During Phase 1, it must be possible to correlate each 3D point (X,Y,Z) in the as-yet unlabeled pointcloud segment to a corresponding pixel (X',Y') on the 2D image collected at the same moment. Although RGB-D sensors would intrinsically provide this 3D-2D mapping, their poor outdoor performance renders them unsuitable for this application. Thus, additional software tools must be developed to achieve the desired 3D-2D correlation. Figure 4 depicts the first step of this correlation process, in which we calculate a pixel-per-degree ratio from the horizontal and vertical field-of-view angular ranges for the RGB sensor, along with the pixel dimensions of RGB frames.

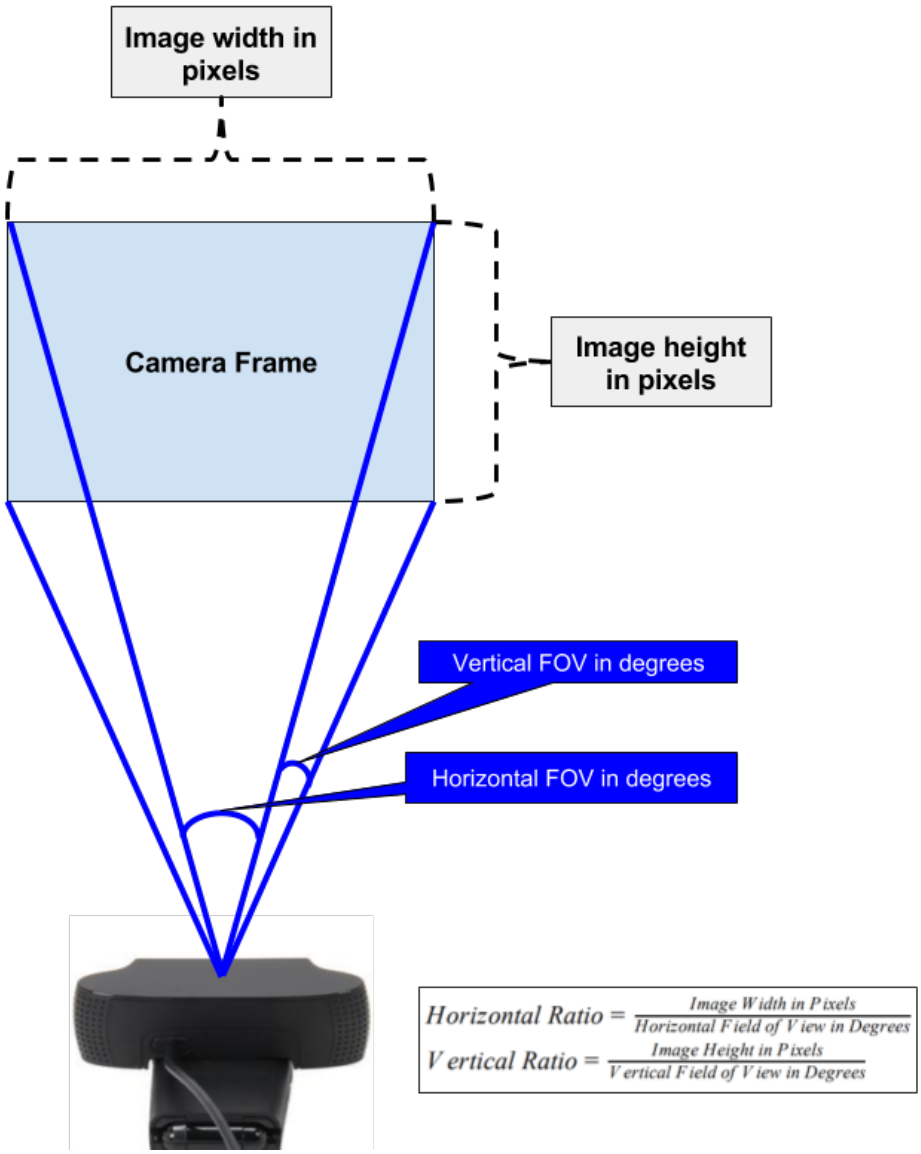


Figure 4. Vertical and Horizontal Pixels-per-Degree Ratio

With the pixel-per-degree ratios pre-calculated, we correlate each 3D coordinate in a pointcloud segment to a pixel location on the synchronized RGB frame according to the process illustrated in Figure 5. For each 3D point in the segment, we calculate the arctangent of the 3D point's Z and Y values and the 3D point's X and Y values. This yields the vertical angular offset and the horizontal angular offset, respectively, from the pointcloud sensor's origin. Multiplying each resultant vertical or horizontal angular offset with the corresponding pre-calculated vertical or horizontal pixels-per-degree ratio yields

the pixel offset from the center of the RGB camera frame, on both the horizontal and vertical planes. As depicted in step 3 of Figure 5, both pixel offset values then undergo a simple 2D translation operation to account for the 2D frame's origin being located in the upper left corner of the frame.

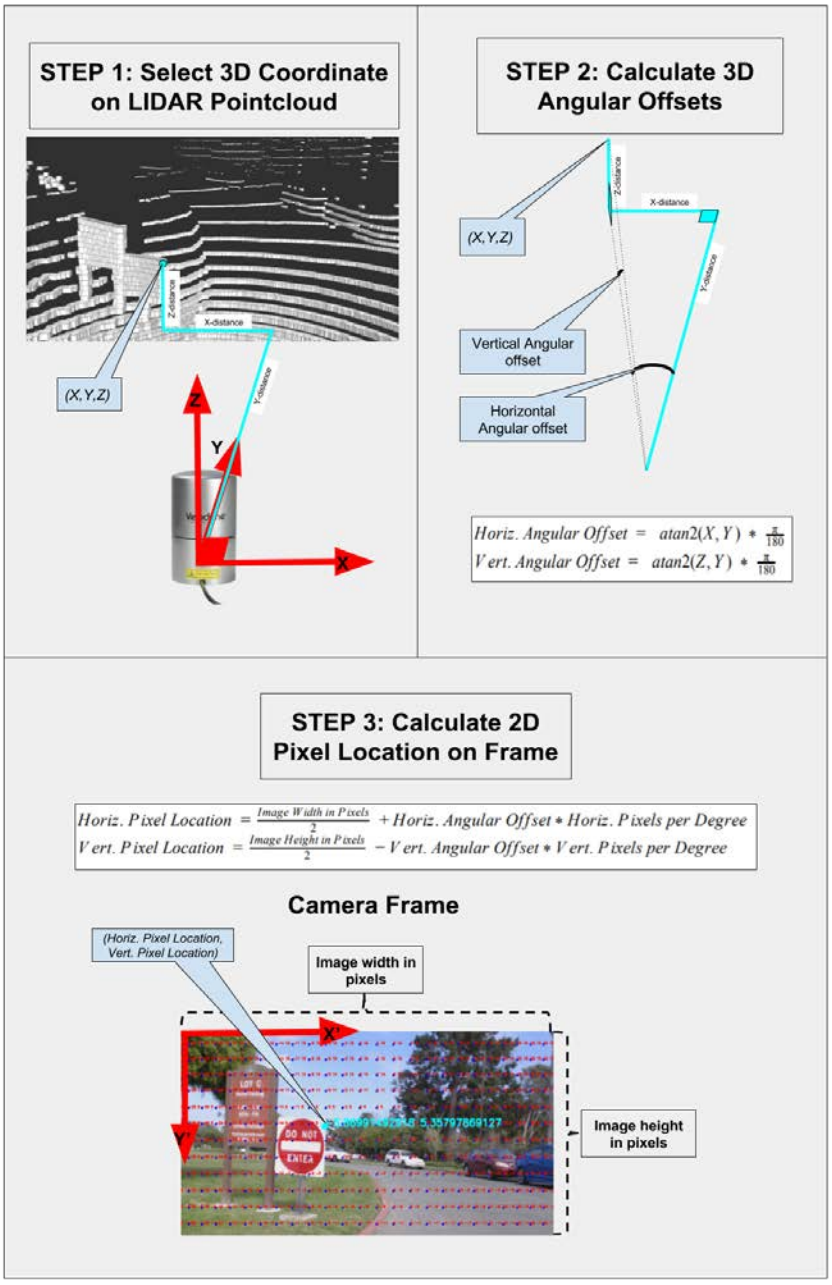


Figure 5. Mapping of 3D Pointcloud Coordinates to 2D Frame Pixels

We iterate through each 3D point in the pointcloud segment and apply this 3D-2D correlation process, ultimately creating a collection of 2D pixel locations that define a bounding box on the 2D camera frame. The 2D image's bounding box, which we use to crop the full 2D image, fully encapsulates the physical object represented with 3D points in the pointcloud segment. To reiterate, this step of Phase 1 ingests a pointcloud segment, calculates the 2D pixel location for each 3D point in the segment, and determines a 2D image crop of the RGB image containing the same physical object represented by the pointcloud segment's 3D points. As detailed in the following section, we subsequently classify the 2D image and apply the resulting semantic label to the pointcloud segment it was derived from.

4. 2D Classification

Previous steps in Phase 1 have shown methodology for producing unlabeled pointcloud segments, and corresponding 2D image crops that represent the same physical object. We now seek to determine a prospective plain English label for the unidentified pointcloud segment. To accomplish this step, we provide the 2D image crop as an input to a 2D image classifier, which outputs a label for the 2D image crop. As contemporary 2D image classification is dominated by neural network-based approaches, we note that these types of 2D image classifiers provide two outputs relevant to our model. First, neural network image classifiers output a list of possible plain English classifications for an input image, typically drawn from a fixed range of class types (e.g. "stop sign") and, second, they output a confidence value associated with each possible classification. We rank order the image classification outputs using their confidence values and the highest confidence result, referred to as the Top-1 result, is the prospective label we apply to our 3D pointcloud segment. However, without any additional filtering, even the Top-1 result may represent a low-confidence identification. This is addressed with confidence-level thresholding.

5. Confidence-Level Thresholding

As of this step in Phase 1, we have produced a prospective plain English label for a pointcloud segment. We enact a simple filtering scheme to discard low-confidence Top-

1 labels. Specifically, if a segment's label is above some user-defined threshold, we continue processing the segment in Phase 1. However, the underlying, now-labeled pointcloud segment remains in the raw data format provided by the pointcloud sensor hardware. These labeled segments are ultimately intended for use in training neural networks capable of classifying pointclouds and this raw pointcloud format is not suitable for input to neural networks. Additional data transformation operations are necessary to convert a labeled pointcloud segment to a representation appropriate for input into a neural network

6. Segment Transformation

Training neural network-based classifiers on pointclouds requires special considerations, particularly with respect to dataset transformation operations. Dataset transformation is a key step in training neural network models and refers to the preprocessing conversion of raw training data to a format suitable for input to a neural network. For example, dominant contemporary neural network models require uniformly sized training set records. For example, when training 2D image classifiers all images might be resized to a fixed pixel height and fixed pixel width [12]. 3D pointcloud classifiers exhibit the same stringent training requirements with respect to needing fixed-size input records. Pointclouds, however—even those representing the same object at different distances—can have vastly different densities. As such, our model applies a series of data transformation operations to the labeled, raw pointcloud segments produced during the previous steps of Phase 1.

Performing a simple re-scaling operation on a 3D pointcloud, similar to those employed on the 2D image datasets, will not reduce the number of X, Y, and Z coordinate values nor address the requirement that training records be of uniform, fixed size. A common, established data transformation operation to resolve this issue is to employ volumetric pixels (voxels) to transform the raw pointcloud data in a labeled segment prior to training a neural network [8]. A voxel is a data structure containing a binary ON/OFF value indicating whether that unit of 3D space is occupied by at least one point in the pointcloud being represented. We propose using a 3D grid of voxels to

represent our labeled pointcloud segment. This grid can be visualized as a stack of chessboard-like layers that divide a 3D space into equally-sized cubes. Under this scheme, any number of points in a pointcloud segment that occupy a given cube (voxel) can be distilled down to a single voxel's binary ON/OFF value, making voxel grids an effective tool for homogenously representing variable-sized pointclouds with a fixed volume. Voxel grids, coupled with 3D scaling operations, allow pointcloud segments representing a wide range of physical objects to be transformed into fixed-sized neural network training inputs (e.g., a 20 x 20 x 20 voxel grid). Figure 6 and Figure 7 depict the data transformation process of converting a raw pointcloud segment to a voxelized representation of uniform volume. This transformation yields a labeled pointcloud segment in a data format suitable for use as a training input to a neural network.

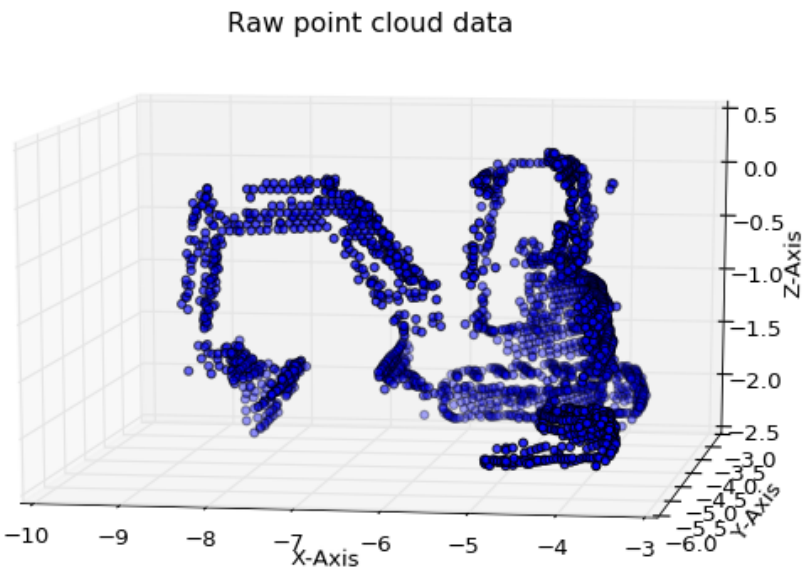


Figure 6. Example of Raw LIDAR Pointcloud of an Excavator from the Sydney Urban Objects Dataset. Adapted from [29].

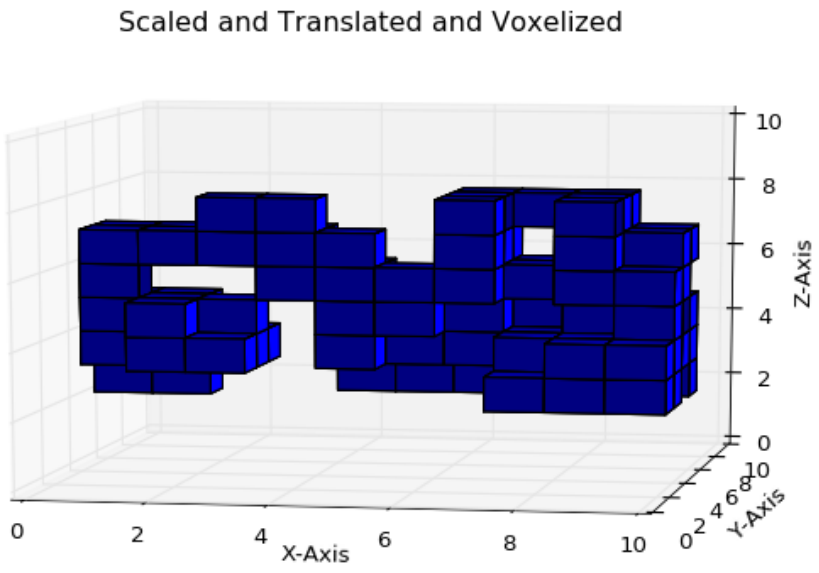


Figure 7. Data Transformation of Figure 6's Excavator to a Scaled, Translated, and Voxelized Representation. Adapted from [29].

7. Neural Network Training

In aggregate, the labeled and voxelized pointcloud segments from the previous step form a dataset suitable for training a neural network pointcloud classifier. We propose following industry-standard best practices for neural network training, to include partitioning 80% of the dataset as a training set and reserving the remaining 20% as a test set to validate the model for accuracy and overfitting. Depending on the requirements of the application, merging similar class types, such as "car" and "truck" into a hybridized "4-wheeled vehicle" class may be appropriate. The training process produces a neural network pointcloud classification model suitable for rapid classification of pointcloud data, which completes Phase 1 of our model.

B. PHASE 2: CONTEXT DISCOVERY

Phase 2, context discovery, is the subject of exploratory research detailed in Chapter three. We provide an overview of the proposed process and then describe the steps for acquiring geotagged media, mapping spatial relationships, and populating a repository to archive these relationships.

Scene recognition is an active area of research in computer vision, and has led to the creation of scene categorization databases, such as the “SceneNet” database [9]. We believe scene recognition could improve the performance of pointcloud classification by factoring in environmental context to weight classification results. Specifically, a database tailored to a specific physical environment and containing the likelihood of an object appearing in this environment, as well as each object’s likelihood of being co-located with other known object types, could serve to boost the confidence value of classification results when operating in said environment. For example, a context database for the Naval Postgraduate School in Monterey, CA, might contain a strongly positive weight for the presence of objects typed “military officer” and might also have positive relationship weights for “military officer” and “flag” appearing in the same scene.

Implementation of this approach requires a relationship database for various object pairings tailored to a specific physical operating environment. The manual creation of such a database would suffer from the same resource-intensive drawbacks as the manual pointcloud labeling that our research seeks to avoid. As such, Phase 2 also seeks to automate the context discovery process, primarily by performing object detection on geotagged media and populating a database containing object type relationships for a given location. We believe our Phase 2 proposal for context discovery, and the production of a corresponding database, could be leveraged to amplify the performance of pointcloud classifiers by applying pre-calculated geographically-localized scene context relationships to object classification results. Figure 8 provides an overview of Phase 2.

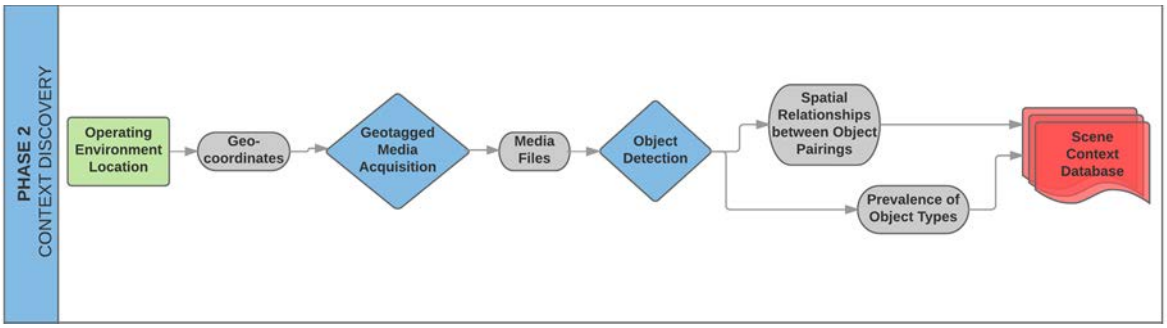


Figure 8. Flowchart of Phase 2, Context Discovery

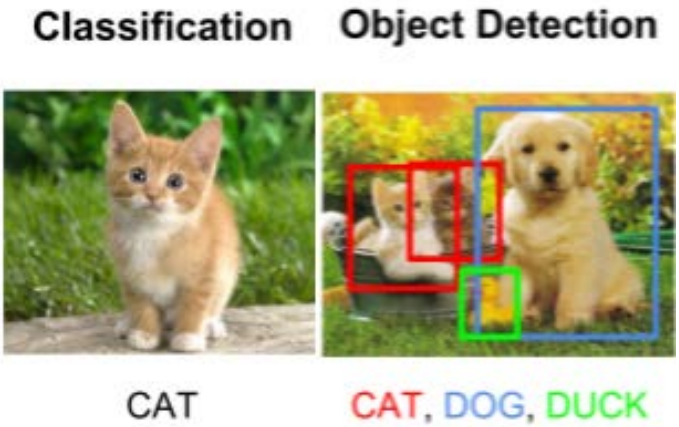
1. Acquisition of Geotagged Media

The first step of Phase 2's context discovery methodology requires foreknowledge of the intended physical operating environment, manifested as a latitude, longitude, and operating radius. Phase 2's context discovery methodology proposes deploying software tools to search the Internet for media, such as photos or videos, containing metadata indicating the media was captured within the desired physical area. Several tools exist for broad Internet searching on specific locational metadata, known as geotags. For instance, both the Bing search engine and YouTube video sharing site provide API access to conduct metadata-based media searching on latitude, longitude, and radius for geotagged image and videos, respectively. During the media acquisition step of Phase 2, we further propose keyword searches, such as "outdoor" or "dashcam," be appended to metadata searches to produce more targeted queries. Metadata-based queries yield links to media files recorded in the desired physical operating environment. Once acquired, each frame of the geotagged media files will be mined for relationship information using object detection tools. This information will then populate the scene context database.

2. Object Detection

In this step, we seek to identify relationships between objects in a given physical environment. To do so, we propose performing object detection on each geotagged media file previously collected in Phase 2. Object detection neural networks provide different output than image classification neural networks. Image classifiers output a single identification for a given image whereas object detection models provide multiple

classifications per frame with bounding box localizations (see Figure 9). As such, we propose processing each geotagged frame through an object detector to identify objects appearing together in a frame. We interpret any resulting per-frame detections to indicate a spatial relationship between those object classes. For instance, a frame that yielded detections of a car, stop sign, and motorcycle would create three relationships, car-stop sign, car-motorcycle, and stop sign-motorcycle. Finally, similar to the previous step of bulk media downloading, the object detection processing can be parallelized and scaled to the level of available hardware processing capabilities.



Classification provides a single identification per image. Object detection provides multiple, localized identifications per image.

Figure 9. Image Classification versus Object Detection. Adapted from [10].

3. Populate Scene Context Database

Our final step of Phase 2's context discovery process is populating a relational database. We propose populating a database with identified relationships between object types, weighted to indicate the prevalence of a pairing's relationship, and a simple absolute count of the hits received by each object type in the geotagged media. As with image classification, object detection neural networks provide a confidence value for each detection and we propose applying a confidence thresholding scheme to these values to limit the insertion of low-confidence, incorrect relationships into the database. To store the relationships, we propose creating an $N \times (N + 1)$ relational database containing a

table and column for each object class, as well as a single static column storing the absolute count of identifications for a given object type. Our current implementation of Phase 2's context discovery algorithm is limited to one specific physical location; however, it could trivially be extended to create relational databases for other locations prior to Phase 3's classifier deployment.

C. PHASE 3: REAL-TIME POINTCLOUD CLASSIFICATION

The previous two phases of our model focused on pre-processing tasks to prepare for deployment of Phase 3's real-time pointcloud classifier. The pointcloud classifier, once trained, has no requirement for an RGB camera sensor and also sheds the requirement to operate in strictly daylight conditions. Ideally, Phase 3 classification operations will employ only a solitary pointcloud-producing sensor and will remain viable in blackout conditions. While our research does not focus on the real-time classification component of the model, we layout a detailed description of our real-time pointcloud classification proposal to provide context and justification for our research. Figure 10 provides an overview of Phase 3.

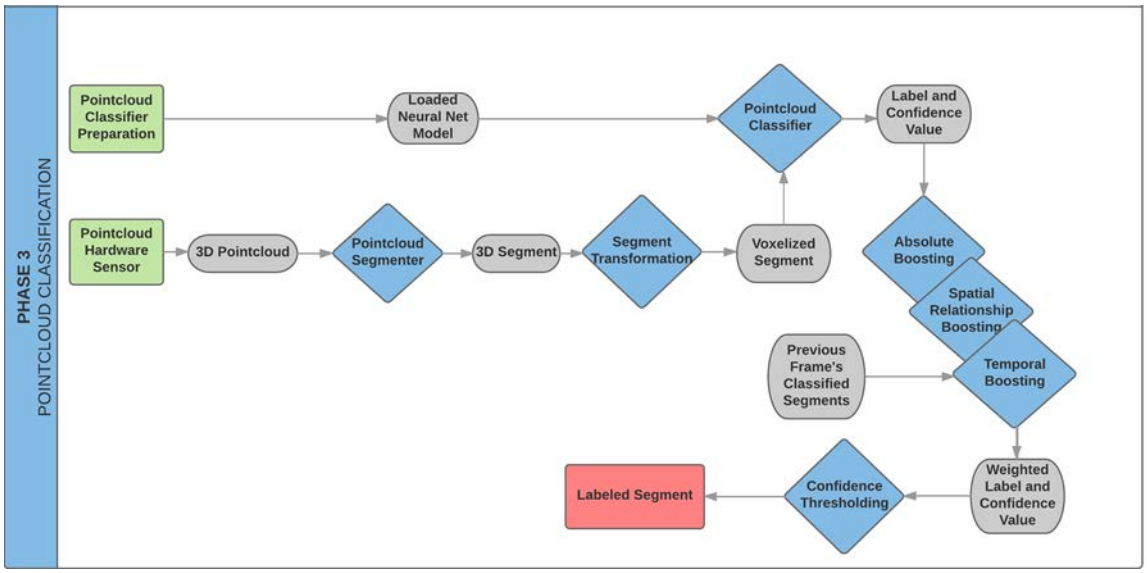


Figure 10. Flowchart of Phase 3, Pointcloud Classification

1. Classifier Preparation

At the onset of mission deployment, we load Phase 1's now-trained pointcloud neural network model into our machine learning toolset's classifier and also load the relational database built in Phase 2.

2. Pointcloud Collection

Phase 3's raw pointcloud collection and processing largely mimics the process detailed in Phase 1, with the exception that requirement for RGB collection in Phase 1 is removed. The RGB sensor's purpose was producing 2D images for labeling the training datasets and, during Phase 3, the classification model has completed its training via those datasets. The removal of the camera requirement likewise removes the 2D-3D correlation step and the requirement to operate in well-lit operating conditions. As pointcloud data is collected by the hardware sensor, it is segmented and voxelized to prepare it for classification with a neural network.

3. Pointcloud Segmentation

Pointcloud data produced by an on-board hardware sensor must be transformed to a format consistent with the inputs originally used to train the pointcloud classification model in Phase 1. As such, we begin with pointcloud segmentation. Unlike during pre-processing in Phase 1, Phase 3 has a real-time requirement, limiting the options for pointcloud segmentation to those capable of achieving near real-time performance. The resultant pointcloud segments from this processing represent discrete scene object and we next mimic the data transformation operations described in Phase 1 to match the format used to train Phase 1's pointcloud classification model.

4. Data Transformation

Phase 3's data transformation operations are the same as Phase 1's, resulting in a fixed-size, voxelized, pointcloud segment representation. With the sensor's pointcloud input transformed into a standardized format, the data is ready for classification by our trained pointcloud classifier.

5. Classification by Neural Network

Following data transformation, each pointcloud segment can be processed by the neural network pointcloud classification model trained during Phase 1. Consistent with Phase 3's near real-time requirement, neural network classification operations are generally dramatically faster than training operations. In this step, each transformed pointcloud segment is processed by the classifier and returns a ranked list of identifications with confidence values. As discussed and noted in Figure 1, even top performing classifiers produce some level of classification error in their results and we explore several approaches to further limiting this error.

6. Results Boosting

To overcome inaccuracies in pointcloud segment classification results, we propose several means of boosting the performance of Phase 3's pointcloud classifier. Up to this point, the classifier's processing of segments evaluated each segment independently. We propose boosting its performance via temporal weighting between subsequent pointcloud frames. Given the highly accurate spatial representation provided by a pointcloud sensor, one could feasibly apply an Iterative Closest Point (ICP) algorithm to correlate current frame segments to high-confidence segments in previous frames, providing a Bayesian prior probability weighting to the classifier's output list when identifying current frame segments. We assess integrating this approach has the potential to boost the accuracy of the classifier.

We also propose boosting performance by leveraging the relational database created during Phase 2's context discovery. Recall that in Phase 2 we produced a database tailored for a specific physical operating environment and containing the incidence rate of two object classes appearing alongside one another in a specific physical environment. We believe it possible to amplify the performance of the pointcloud classifier by applying the pre-calculated, geographically-localized scene context relationships in Phase 2's database to our real-time pointcloud classification results.

We propose two approaches to leverage Phase 2's database. First, we look to apply static weights to the entire classification result list. Recall in Phase 2 the relational database maintained an absolute hit count for each object type detected in media geotagged to the operating environment. The absolute hit counts, normalized across all class types to sum to 1, are an indicator of the prominence of each object type in the operating environment. We add each normalized absolute count as a weight to its corresponding class type's confidence value in the classifier's result list. This weighting boosts the classification confidence of object types known to be present in the environment and has limited effect on those not encountered during Phase 2's context discovery. Similarly, we propose adding normalized instance counts of scene pairings, identified during Phase 2, as weights to the classifier's results list. Recall that in Phase 2 our model identified spatial relationships between object types based on their co-location in geotagged video frames. We now apply those identified relationships to the real-time classifier's results list. We intend to boost an identification's confidence value based on other high confidence identifications in the scene and any spatial relationship recorded during Phase 2.

For example, a pointcloud scene might contain several segments, with some receiving high-confidence classifications prior to any boosting. For scene segments with weaker confidence values, which might otherwise get filtered, we propose conducting a database lookup on the object types of the scene's high confidence identifications, yielding an array of predetermined spatial relationships for the scene's high confidence segments. These relationships, which contain the prevalence of two object types appearing in the same frame, represent a weighted spatial relationship. We add these weights to the unidentified segment's classification results to boost the confidence values of its results, based on its known spatial relationships with other scene objects. For example, if a scene has a high confidence classification for an object of type "car," each predetermined spatial relationship for type "car," such as "stop sign," "pedestrian," and "truck," would be boosted in the results list of any segments below the confidence threshold in the scene. In this example, a scene segment with results "baseball bat" and "stop sign" in its results list would have its "stop sign" result amplified based on "car's"

predetermined relationship with “stop sign” and the high-confidence identification of a “car” as another segment in the scene.

7. Confidence Thresholding

Our final proposed step of Phase 3 is to enact a confidence thresholding scheme similar to that seen in Phase 1. Despite efforts to boost our classifier’s accuracy, some results will remain low-confidence identifications and we seek to avoid providing these identifications by dropping any identification whose top-1 result falls below a selectable threshold.

D. MODEL CONCLUSIONS

The three phase model defined earlier provides a comprehensive, novel framework for pointcloud classification. We presented a methodology for automating the creation of training data for a pointcloud classifier (Phase 1), mapping out spatial relationships between object pairings (scene context) in an operating environment (Phase 2), and integrating training data and scene context into a real-time pointcloud classifier (Phase 3).

III. RELATED WORK

We begin with a review of related works in the field of neural networks and pointcloud classification, followed by a more detailed review of related works and developments in the field of dataset creation for neural networks.

A. NEURAL NETWORKS

The theoretical basis for neural networks was first devised over 70 years ago [11]; however, groundbreaking research on deep neural networks in 2012 [12] were responsible for triggering drastic advances in neural network-based image classification performance over the last several years (Figure 1). These advances have been demonstrated at the annual ImageNet Large Scale Visual Recognition Competition (ILSVRC). One notable deep learning neural network model, Inception-v3, which won the ILSVRC 2014 competition [13], was trained on the 1000 object classes in the ImageNet photo dataset and currently boasts a 3.46% Top-5 error rate on the ImageNet validation image set [14]. More plainly, 96.54% of the time the Inception-v3 model will output the correct classification "guess" as one of the top 5 classification matches, when tested against the ImageNet validation set of images. Following these well-known advancements in 2D image recognition, deep neural networks were adapted to perform object recognition on 3D pointcloud data. Most notably, the VoxNet team developed techniques for transforming pointclouds into a data representation suitable for processing by neural networks [8]. VoxNet's technique transforms each object's pointcloud data into volumetric pixels (voxels) (see section 3.B.2) and represents each object in a fixed size 3D occupancy grid. However, VoxNet's reliance on voxels to train neural networks on pointcloud classification requires raw pointcloud scenes first being parsed into discrete objects, or segments. The availability of tools offering pointcloud segmentation via Euclidean clustering [15] and depth clustering [16], [53] provide the ability to filter raw pointcloud scenes down to segmented representations of discrete objects, suitable as inputs to neural network-based classifiers. We found that Depth Clustering generally produced accurate segments while segmenting LIDAR scenes imaged in outdoor

environments lacking densely placed objects. Fast-paced advances in creating the pointcloud training datasets required by these neural networks, however, have not manifested [17] and are the focus of our research.

B. DATASET CREATION

Current techniques for labeling neural network datasets typically involve either bulk manual labeling by humans, commonly via crowdsourcing, or leveraging custom tools to assist humans in the tedious process of hand labeling. We first provide a discussion of hand annotation of datasets and the difficulty of creating datasets without the help of tools, and then explore leveraging software tools to increase the throughput of creating annotated training datasets. Our research focuses on automating the latter approach and, while there are no examples of fully automated LIDAR dataset creation pipelines available, we discuss other research in producing synthetic datasets that could feasibly be automated. We specifically highlight closely related work in [17], which clearly identified the difficulty of creating labeled 3D pointcloud data, noted that the lack of such data was a "bottleneck" in the advancement of 3D object recognition, and provided innovative techniques for pointcloud labeling.

1. Hand Annotated Datasets

The predominant approach to creating accurate, labeled datasets remains manual, human-involved labeling. Many historic and contemporary datasets, to include the MNIST handwriting sample dataset and the Pascal VOC dataset were created with human-involved labeling [18]. Historic datasets, such as the venerable Iris Flower dataset containing 150 samples [19], were the result of laborious hand labeling by a trained expert; however, advances in fee-for-service Internet crowd-sourcing tools, such as Amazon's Mechanical Turk, have allowed for distributed creation of much larger labeled datasets by a non-expert human labor pool [20]. Unsurprisingly, significant quality control issues can arise when moving from human expert labeling to a distributed crowd-sourced approach [21]. COCO, a dataset comprised of over 200,000 segmented and labeled images [22], and ImageNet, a dataset containing over 14 million labeled images [23], were created by employing a large human talent pool of annotators via the

distributed crowdsourcing tool Mechanical Turk. Yet, despite crowd-sourcing its annotations, ImageNet boasts 99.7% accuracy with its labels. ImageNet achieved this accuracy by requiring multiple annotators to submit labels for a given image and then taking a majority vote from the results, limiting the insertion of incorrect labels into the dataset [24].

The effort required for human labeling of large datasets is substantial, as evidenced by the stringent and complicated guidelines issued to human annotators of the Pascal VOC dataset [25], and remains a major barrier to new dataset creation and, subsequently, the training of neural networks on new problem sets. Pointcloud datasets, compared to 2D image-based datasets, require even greater effort to create, even when collected under controlled, indoor conditions with RGB-D cameras. For instance, the "RGB-D Objects Dataset," characterized as a large dataset with over 300 household objects, was created by humans placing each object type on a rotating turntable for imaging and then hand labeling the resulting pointcloud [26]. The BigBird dataset also imaged objects with a turntable and effectively reduced the annotation time per object from 20 minutes down to five [27]. Other approaches, such as the "The Large Dataset of Object Scans" project, employed a team of 70 human operators with similar mobile scanning hardware to create a dataset of 10,000 pointcloud scans [28]. Clearly, the requirement to physically position an object on a turntable and waiting for it to rotate 360-degrees to create a 3D pointcloud represents a significant expenditure of effort, particularly when compared to the relatively simple process of labeling a 2D image by drawing a polygon bounding box.

2. Tool-Based Annotation of Datasets

The Sydney Urban Objects Dataset is a popular, segmented LIDAR pointcloud dataset consisting of only 588 records, all of which were meticulously hand annotated [29]. Ongoing research seeks to create toolsets that lessen the burden of labeling such 3D pointcloud datasets [30], potentially enabling the creation of larger datasets. One industry leader in the creation of labeled datasets for autonomous vehicles, Mighty AI, has developed a slew of development tools to increase the throughput of human annotation

across a range of dataset types, and has noted costly annotation times of approximately one worker-hour for full-frame, per-pixel outline segmentation and semantic annotation of a 2D image when selecting labels from a list of 75 classes [31]. Clearly, the expenditure of one worker-hour to hand label a single frame illustrates the difficulty of scaling some dataset creation tasks without tremendous allocation of resources, further demonstrating the need for automated dataset creation tools. One such tool, LabelMe, is an open source annotation tool that provides a polygon-drawing capability for human beings to segment and label images to a preselected range of classes [32]. Our research is complementary to these tools and techniques and seeks to improve upon them by further automating the annotation process.

Hackel et al. [17] made a large dataset of labeled LIDAR data available under the Semantic3D.net project featuring a total of eight classes and developed two interesting tool-based approaches to label their large LIDAR dataset. For annotation in a 3D viewing environment, their approach required a human annotator to select only a small number of points from a desired object and then, once a predefined model type is selected, grows the selected pointcloud to contain surrounding points that also fit the model's parameters [17]]. The model, once fit to the object and pruned for outlier points, was successfully employed by human annotators to segment objects and was able to "select large buildings in a couple of seconds" [17]. The Semantic3D team developed another tool-based approach to annotating that provided for easier navigation by the human annotator [17]. The technique involved the human annotator first selecting a camera viewpoint within the 3D viewing environment, casting that viewpoint to a 2D viewpoint, and drawing a 2D bounding box around the desired object [17]. This process was repeated by the human annotator until the resulting intersection of these bounding boxes contained only inlier 3D points for the desired object [17]. The annotator would then assign a class label to the segmented points. This tool-based segmentation is slower than the Depth Clustering algorithm, but considerably faster than manually selecting individual points for segmentation.

3. Synthetic Dataset Creation

While we are not aware of a fully automated pipeline for labeling LIDAR datasets, we highlight other work capable of creating labeled datasets with limited human involvement, albeit with synthetic vice real-world datasets as input. Virtual 3D environments containing labeled 3D models provide opportunities for synthetic dataset creation. For example, the SceneNet project is capable of producing a nearly infinite amount of synthetic labeled 2D scene records [33]. SceneNet's approach is to model an artificial 3D environment or scene, hand annotate the 3D objects as a one-time cost, and then create a nearly limitless amount of 2D representations of the labeled "scenes" by varying the location of a virtual camera to rasterize the frame [33]. While a different approach than our proposed pipeline, we highlight this approach as a technique for creating labeled datasets in a nearly automated fashion.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. BACKGROUND

Following the application of LIDAR for obstacle avoidance in the 2005 DARPA Grand Challenge, significant progress has been made in expanding the application space of LIDAR data via neural network-based object classification [3], [8], [34], [35]. As our research aims to advance the field of dataset creation for these LIDAR classifiers, we first provide an overview of the underlying technology, namely neural networks. We continue this discussion with an overview of pointclouds and their representation as voxels when processed by neural network-based classifiers.

A. NEURAL NETWORKS

We begin by providing an overview of neural networks themselves, followed by a discussion of TensorFlow and Inception, the specific machine learning toolset and pre-trained model we employ during our Phase 1 implementation (see Section V).

1. Overview of Neural Networks

Machine learning-based computer vision has made tremendous strides over the last decade, largely through the application of neural network image classifiers [36]. The mathematical details of neural networks are beyond the scope of this research; however, a brief overview of their inner workings, capabilities, and training requirements is appropriate due their centrality to our research.

Neural networks are defined as “a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs” [37]. These processing elements are referred to as “nodes” and can be grouped into “layers” if a collection of nodes all operate on inputs from a previous layer and do not operate on inputs emanating from nodes within their own layer. At a minimum, neural networks typically have an input layer, which ingests external data to operate on, a series of internal “hidden” layers that have no external interface, and an output layer [38]. The output layer is responsible for ultimately providing the neural network’s response to the external input and, in the case of image

classification neural networks, may contain one output node for each object class that the network is capable of classifying [39]. The output nodes each produce a numeric response to the given input, representing the output class's correlation with the provided input. Further, neural networks generally require each input data record to be of uniform length which commonly requires data transformation operations be applied to datasets containing variable length records (see section 3.B.2). Figure 11 depicts a simple neural network.

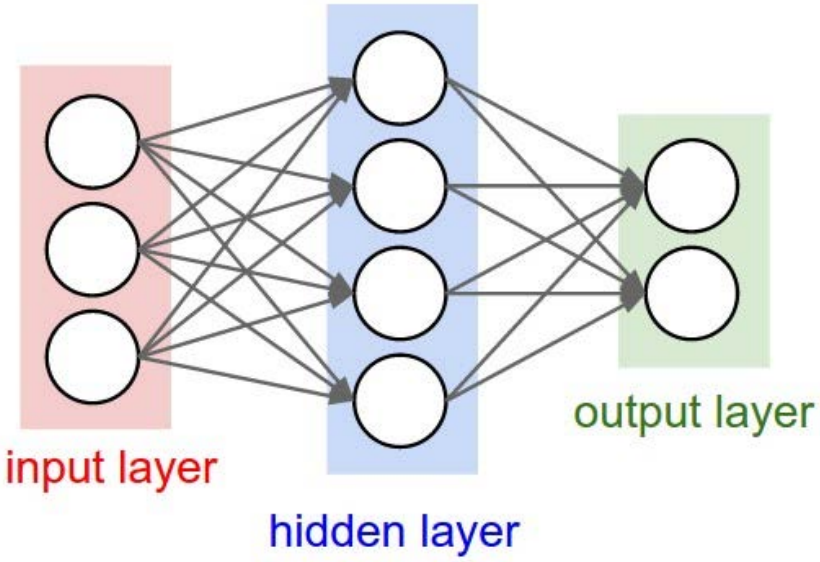


Figure 11. A Simple Neural Network. Source: [40].

Neural networks, particularly "deep" neural networks involving many layers, can require a large amount of labeled training data ahead of time to produce accurate outputs during operation. For example, the 22-layer GoogLeNet model was trained on 1.2 million labeled images [39]. A trained neural network is capable of ingesting an unlabeled photo and outputting a ranked list of classifications for the content of the photo, alongside a confidence value reflecting how consistent the unlabeled photo is with a corresponding classification. The classification space is limited to the range of object classes (e.g. "car," "dog," etc.) that the neural network was trained on so, clearly, a neural network can only identify those objects that it was specifically trained to identify. Furthermore, this training

data must be labeled (e.g., a photo of a dog must be labeled “dog”) ahead of time to allow for the neural network to learn the association between the pictured object and the output class. Creation of this labeled training data (“annotation”) remains a critical bottleneck in creating effective neural network-based classifiers and is described further in subsequent sections.

2. TensorFlow and Inception

Our research and proposed dataset creation pipeline relies on the machine learning toolset TensorFlow and the pre-trained neural network model Inception. TensorFlow is an open source, machine learning toolset developed by Google [41] and is the fastest growing tool for machine learning [42]. While it is possible to train neural networks “from scratch” with TensorFlow, our research creates labels from the output of a popular pre-trained image classifier model known as Inception, specifically, the third revision known as Inception-v3. Our pipeline’s image classification is conducted with an unmodified version of Inception-v3;

B. POINTCLOUDS

As our research focuses on dataset creation for LIDAR data classifiers, we provide an overview of pointclouds, a common data format for representing physical, LIDAR-imaged objects. We follow this overview with a detailed description of voxels and discuss the practice of voxelizing raw LIDAR pointclouds prior to training pointcloud-classifying neural networks.

1. Pointcloud Overview

A pointcloud is a collection of data points in a coordinate system, typically represented as X, Y, and Z coordinate values which depict the outer surface of an object [43]. In our pipeline, these values are represented using an East North Up (ENU) reference frame originating at the LIDAR’s laser (Figure 12).

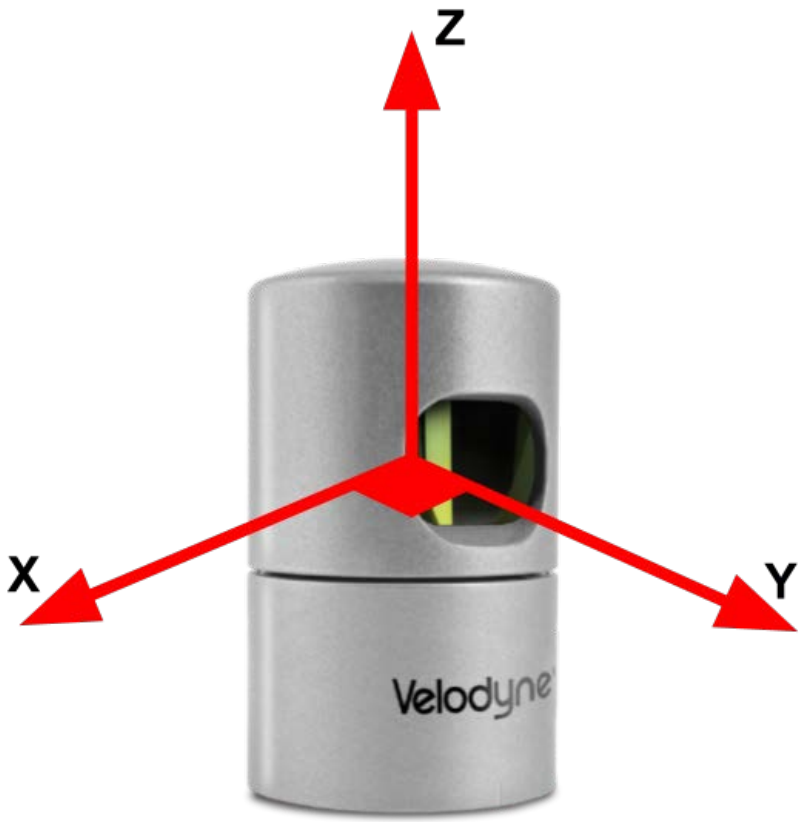


Figure 12. East-North-Up LIDAR Reference Frame Defining Pointclouds Coordinates

Our research focuses on real-world pointclouds depicting physical objects, produced by a range of hardware sensors, including LIDAR, sonar, radar, stereo cameras and RGB-D cameras. While providing potentially highly-accurate spatial representations of objects, pointclouds collected by these sensors typically provide a low resolution representation of their surroundings than modern camera images. For instance, the highest fidelity LIDAR available, the Velodyne HDL-64e, which is largely reserved for research applications [44], provides a mere 64 lines of vertical resolution [45] whereas modern 4K cameras provide 2160 lines of vertical resolution. While limited in angular resolution, the pointclouds produced by this sensor are accurate to within 1.5 centimeters for objects in a 360-degree horizontal field of view [46]. These inherent features of LIDAR data require a unique approach when training neural networks to identify objects in LIDAR data.

V. METHODOLOGY

We focus our research's implementation and experimentation on Phase 1 of our proposed model, automated dataset creation. We also conduct exploratory work on Phase 2 of the model, context discovery, and provide our initial findings in support of future work.

A. PHASE 1 IMPLEMENTATION

We implement the four steps of Phase 1's automated dataset creation by developing a pipeline architecture that conducts, in the following order, pointcloud segmentation, 3D-2D correlation, 2D classification, and confidence-level filtering. Our Phase 1 implementation is designed to produce labeled, high-confidence pointcloud segments without human intervention during the annotation process. For example, under ideal circumstances, the pipeline could output a cluster of laser returns (a pointcloud segment) containing the 3D representation of a stop sign and automatically label this collection of points, "stop sign." More specifically, the pipeline's implementation accomplishes this by ingesting synchronized LIDAR and camera data as inputs and outputs labeled pointcloud segments as '.pcd' files. Our implementation's approach to each step of Phase 1 will be discussed in the following sections and we note that the final step of Phase 1, training a neural network on the resultant dataset, is not handled by our pipeline or evaluated in our research. Figure 13 provides a graphical overview and represents our implementation of the Phase 1 model portrayed in Figure 3.

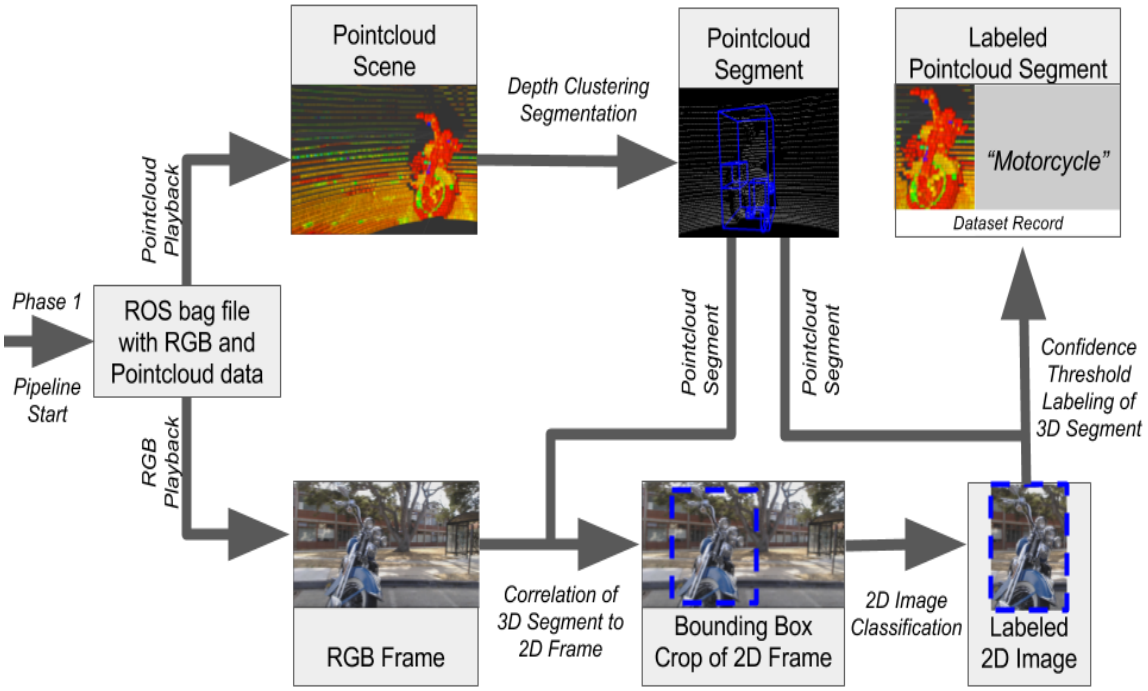


Figure 13. Implementation of Phase 1's Automated Dataset Creation

1. Synchronized Collection of Pointcloud and RGB Data

The proposed processing pipeline for dataset creation requires the synchronized collection and fusion of 2D imagery with accurate 3D pointcloud data. While a variety of RGB-Depth sensors such as the Microsoft Kinect, known as "structured light sensors," are well suited for the calibrated fusion of 2D imagery on 3D pointclouds, these sensors suffer from short maximum range as well as limited usefulness in brightly lit outdoor environments due to sun glint interfering with the depth sensor [47]. Further, affordable stereo cameras offering 3D pointcloud creation in outdoor environments and featuring small inter-ocular distances, such as StereoLabs ZED camera, are available, but do not offer the range of a Velodyne LIDAR sensor [48]. As such, the highly accurate, long-range Velodyne HDL-32e was selected for 3D pointcloud creation and paired with an array of Logitech c920 RGB cameras. This configuration required the fabrication of a custom hardware collection platform and several open source software tools to conduct data collection. Figure 14 depicts a visualization of the raw data collected during this

phase. Our pipeline loaded a manufacturer-provided calibration file for the LIDAR and did not correct for lens distortion on the c920. Synchronization of LIDAR and RGB data remains a concern for our implementation and future revisions may improve classification accuracy by replacing the c920 array with a calibrated, 360-degree camera such as the Ladybug®5 spherical imaging system.

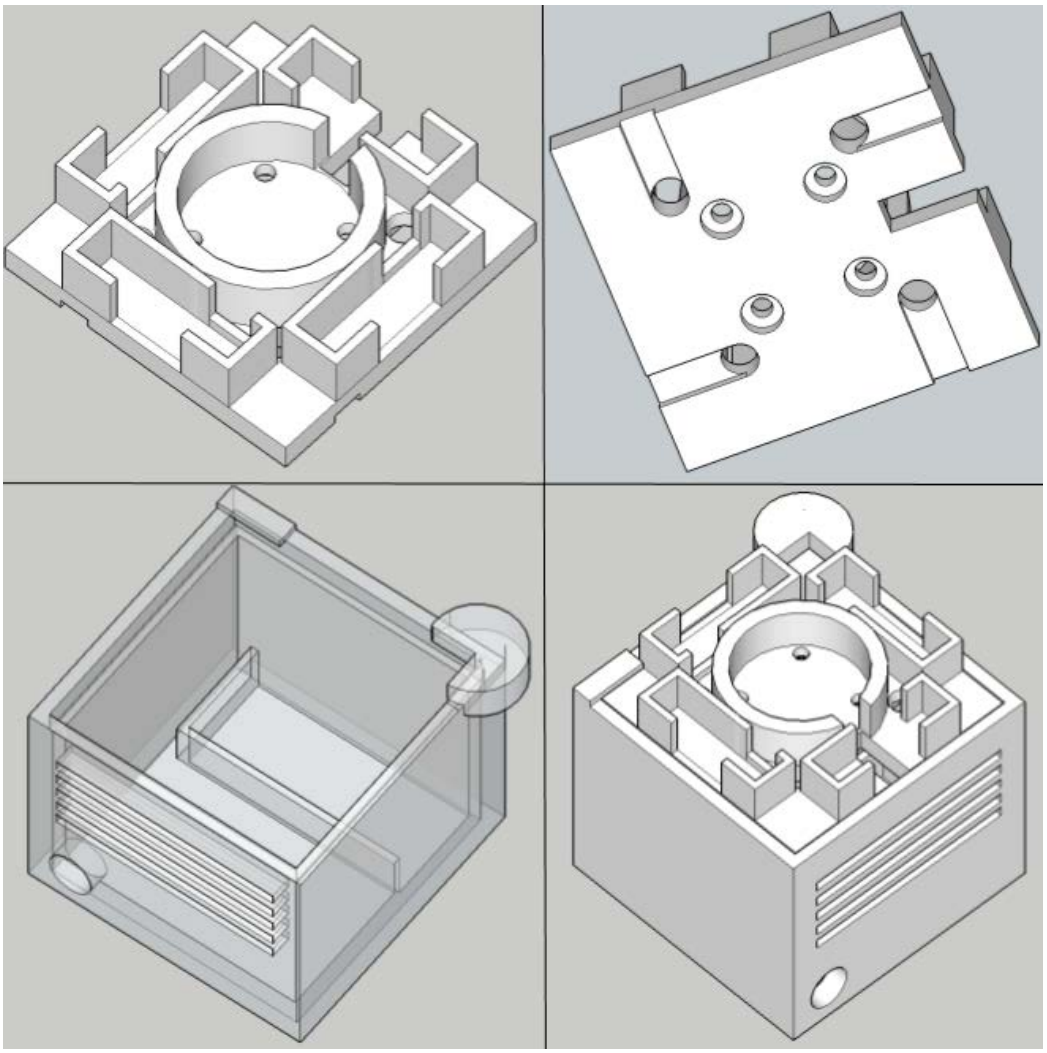
Figure 14. Visualization of LIDAR Pointcloud and Camera Data Collected

a. Tandem LIDAR+CAMERA Mount and Mobile Collection Unit

We designed a mount plate with CAD software to hold the Velodyne HDL-32e scanning laser (LIDAR) and three Logitech c920 HD cameras. The three cameras are oriented at 90-degree offsets on the horizontal plane. A mobile collection box was also designed to house a 12V 9Ah lead acid battery, a Cyberpower 200W inverter, and a Velodyne LIDAR interface box featuring the LIDAR's input/output. These components are depicted in Figure 15. A standard CAT5 Ethernet cable was used to transmit data from the interface box to our Dell XPS 9650.

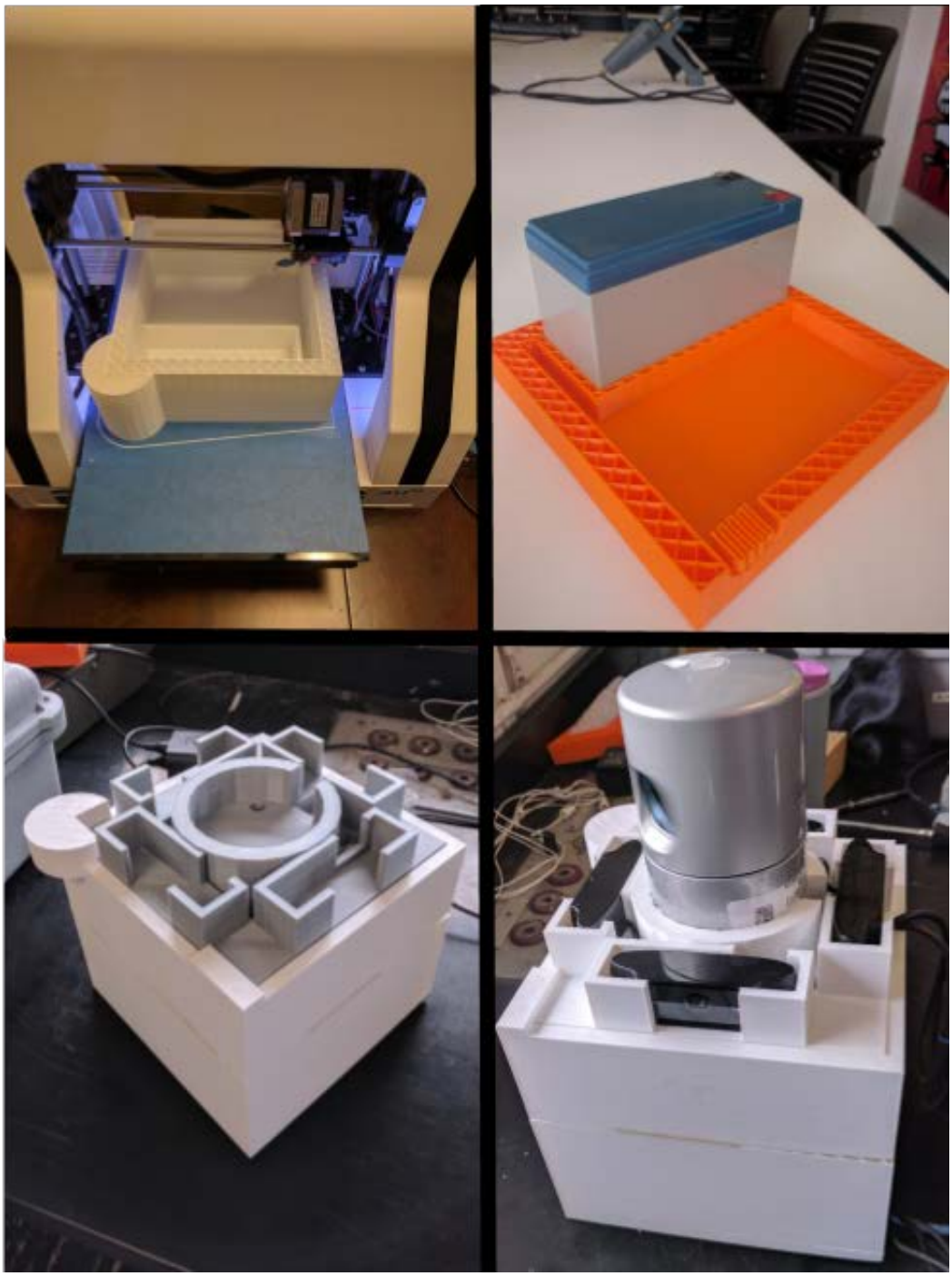
Figure 15. From Left to Right, Top to Bottom: Cyberpower 200W Inverter, 12V Battery, Velodyne LIDAR Interface Box, and Fully Loaded Mobile Collection Box

These two mounting accessories were 3D printed in PLA filament with 10% infill and attached to the roof of a passenger vehicle for data collection. Our CAD and 3D printable stereo lithography files have been made available under Appendix, Sections A-D with an open source license. A CAD representation of the mobile collection box is depicted in Figure 16 and our fabrication process is depicted in Figure 17.



From top left, top of mount, bottom of mount, internals of collection box, mount attached to mobile collection box

Figure 16. Collection Mount and Mobile Collection Box Design



From upper left: printing mobile collection box in PLA with 10% infill, cutout of mobile collection box showing battery tray, mount latched to mobile collection box, full collection system with sensors

Figure 17. Collection Mount Fabrication Process

In addition to the LIDAR and three HD cameras, data was collected with a Dell XPS15 9650 featuring a solid state drive (SSD), as well as a USB-C to USB-A dongle to enable simultaneous collection from all three cameras. The higher I/O capabilities offered by the SSD, relative to external USB hard drives with magnetic storage, were required to avoid dropping data during collection. While the physical mount plate can accommodate a total of four HD cameras, all data collection was limited to three cameras due to firmware-imposed restrictions on the maximum number of HD cameras per USB bus. Specifically, the c920 firmware does not allow more than one camera per USB bus and will abort attempts to initiate a camera stream when more than one camera is detected per USB bus. Our collection laptop had three USB buses which limited our collection to three simultaneous c920 camera streams. This restriction could be overcome in future pipeline iterations by replacing the c920 cameras with a single 360-degree camera.

b. Collection Details

Data collection was conducted in and around the Naval Postgraduate School campus in Monterey, California and on San Clemente Island, California. All data was collected with the tandem LIDAR and camera collection rig. The rig was mounted on the roof of a passenger vehicle and all data was collected while driving at an estimated maximum speed of 30 mph.

c. LIDAR Hardware

All pointcloud data was collected from a Velodyne HDL-32e LIDAR, a sensor capable of producing 700,000 3D points per second [49]. Data from the LIDAR was produced at a rate of 10Hz and was collected at a minimum range of 0.9 meters and maximum range of 130 meters.

d. Camera Hardware

2D camera data was collected from three Logitech c920 HD cameras oriented 90 degrees apart. As the scanning LIDAR provides a 360-degree view of scene, the data processing pipeline could feasibly accommodate a full 360 degree 2D view as well. However, as indicted, data processing issues limited the collection to only three

simultaneous camera feeds. Our tandem collection setup further limits the horizontal and vertical field of view (FOV) available to the pipeline. The c920's horizontal FOV is approximately 70.42 degrees [50] leaving a nearly 20-degree gap between adjacent c920 cameras that is not collected. Figure 18 depicts the effective horizontal FOV of our collection setup.

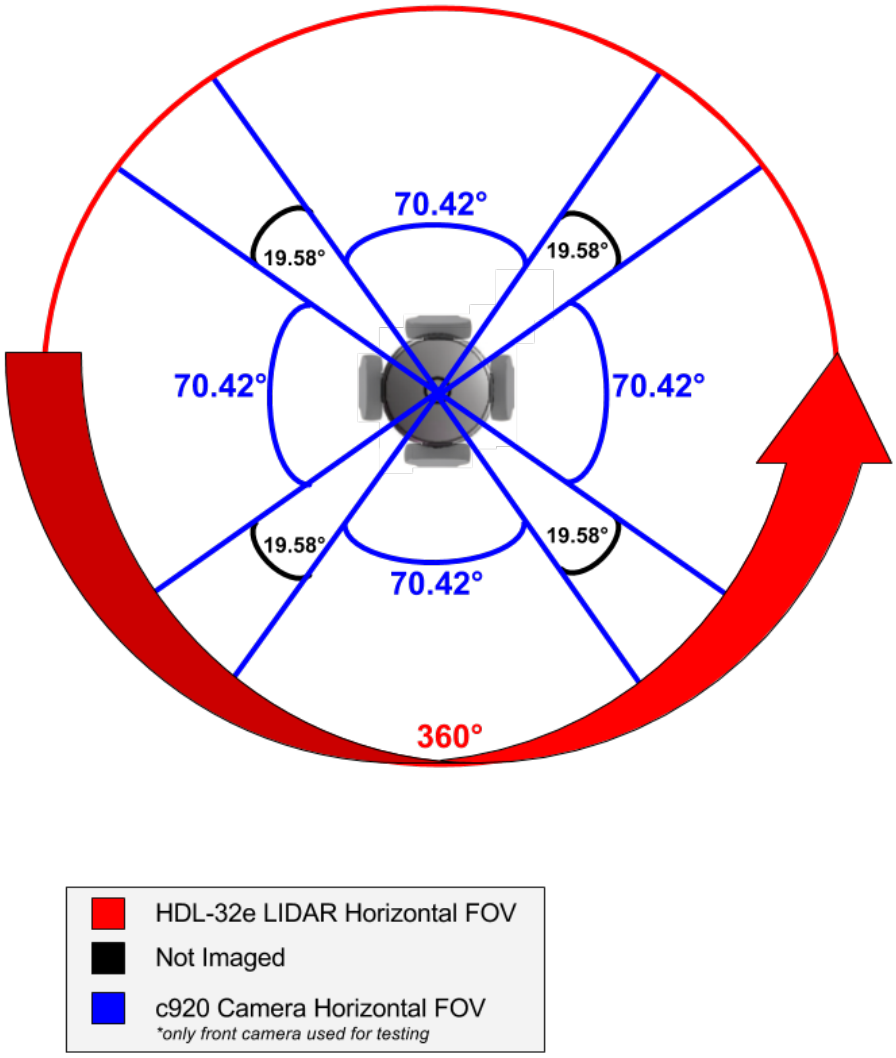


Figure 18. Horizontal Field of View of Camera and LIDAR Collection Rig. Only the Front Camera's Collection Was Used for Testing.

The c920's vertical FOV is approximately 43.30 degrees and the LIDAR's vertical FOV is 40 degrees, but the LIDAR's vertical FOV is angled downward and ranges from +10 to -30 degrees. The overlap between these two sensors provides an effective vertical FOV of approximately 32.32 degrees (see Figure 19). Our pipeline addresses the inconsistent FOV capabilities of the two sensors by ignoring segments containing points that map to pixel locations beyond the camera's horizontal or vertical FOV. This approach has the effect of missing opportunities to process segments as they transition out of the camera's view.

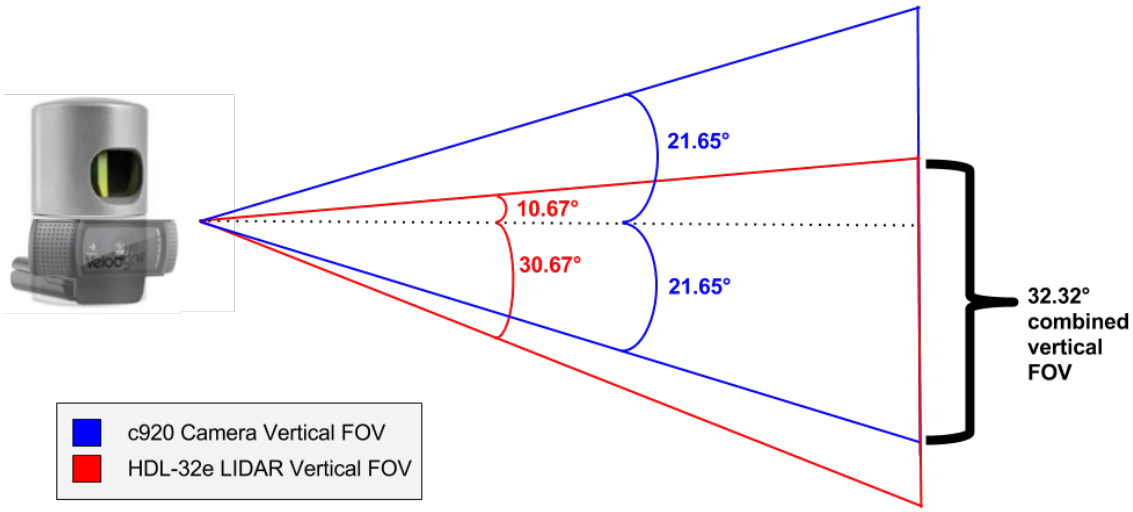


Figure 19. 32.32 Degree Vertical Field-of-View Available to Phase 1 Pipeline

e. Robot Operating System nodes

The Robot Operating System (ROS) provided the software infrastructure for processing and storing our LIDAR and camera data in a synchronized manner. ROS is a publish-subscribe message passing architecture designed for robotics and features a multitude of integrations for computer vision and pointcloud processing [51]. In addition to the core ROS infrastructure provided in all ROS applications, two additional ROS nodes were employed - usb_cam and velodyne_pointcloud. ROS nodes are parallel processes that communicate with a central ROS master node and typically pass messages

to each other. Three instances of the usb_cam node were run during data collection to process three simultaneous streams of camera frames, with each video stream encoded with MJPEG compression at a resolution of 1920 x 1080 at 30Hz. RGB frames from the three cameras are published as ROS “sensor_msgs/CompressedImage” messages under the ROS topics “/usb_cam1/image_raw/compressed,” “/usb_cam2/image_raw/compressed” and “/usb_cam3/image_raw/compressed.”

Additionally, a velodyne_pointcloud ROS node ingested raw UDP data packets from the LIDAR and publish the data as ROS PointCloud2 messages on the topic “/velodyne_points.” Raw packets were also preserved in the data collection as ROS “VelodyneScan” messages and published under the topic “/velodyne_packets.”

Archiving the synchronized LIDAR and camera data was done via ROS bag files. Bag files store specific ROS topics to disk as time series data and, in the case of the topics published for the LIDAR and three camera streams, the bag file archives grow at a rate of 2 gigabytes per minute. All bag file datasets are linked in the Appendix and can be played back in ROS for visualization in RViz.

Figure 20 details the data collection process for creating synchronized LIDAR and camera datasets. Further, to expedite our evaluation of the Phase 1 implementation, we applied the velodyne_pointcloud node’s reprocessing functionality on the original LIDAR data. This reprocessing culled the LIDAR data to a maximum range of ten meters. We further configured our implementation to only create labels for pointcloud segments in the field of view of the front facing camera, further simplifying our evaluation in Section VI. These limitations can be removed in future implementations to achieve a wider field of view.

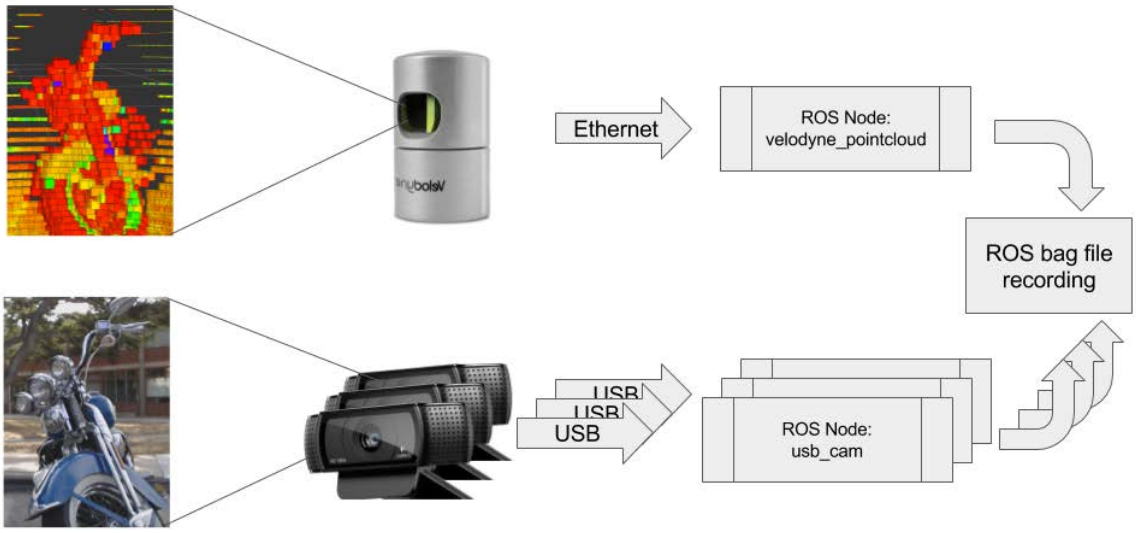
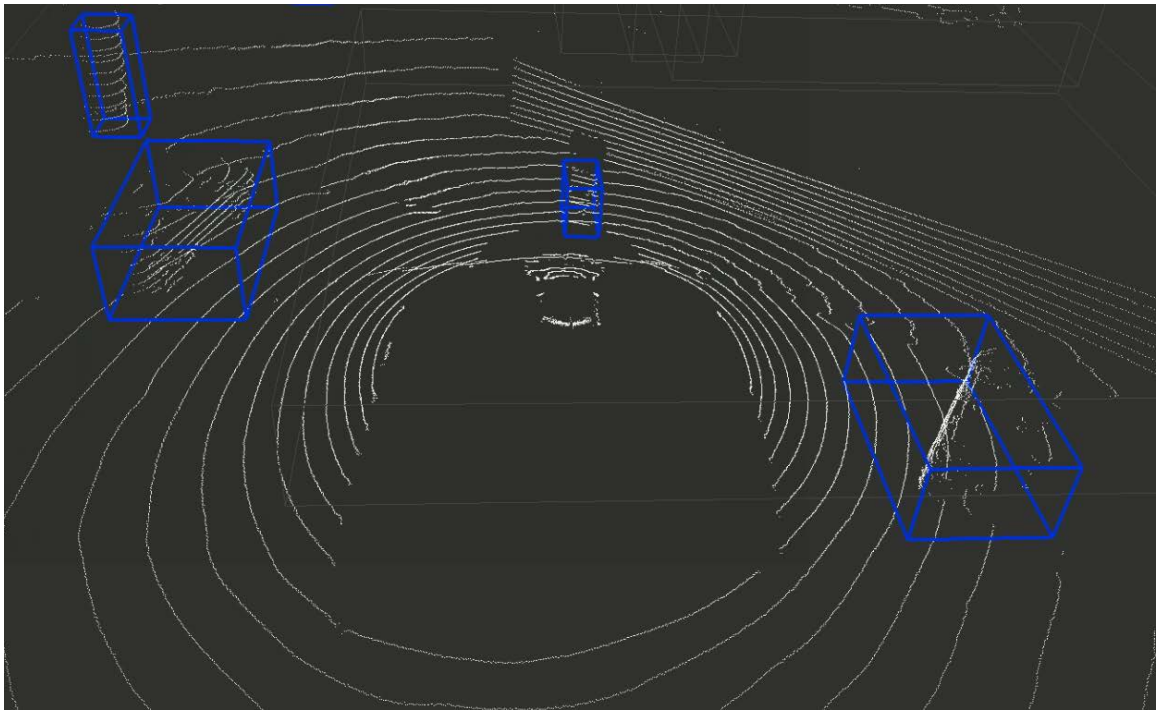


Figure 20. Collection of Synchronized LIDAR and Camera Data

2. Pointcloud Segmentation

For our pipeline, two segmentation approaches were evaluated: Difference of Normals (DON) segmentation and Depth Clustering segmentation. Difference of Normals segmentation, a technique implemented in the Point Cloud Library (PCL), combines surface normal calculation with Euclidean clustering to segment a 3D pointcloud [52]. This approach requires computationally intensive 3D vector math and took more than ten seconds to segment a single LIDAR frame from the HDL-32e with the Dell XPS15 9650's Intel Core i7 CPU. Depth Clustering avoids the computationally intensive 3D vector math of Euclidean Clustering-based DON segmentation and, instead, creates 2D range images from LIDAR-derived depth values [53]. Depth Clustering segmentation also boasts a 1000x speedup over Euclidean Clustering methods [53] and allows for real-time segmentation of pointclouds from the HDL-32e on our development platform. More importantly, Depth Clustering segmentation was shown to produce pointcloud segments of similar quality to those produced using Euclidean Clustering [16]. Therefore, we opted to Depth Cluster our raw LIDAR scenes into segmented pointclouds.

During the segmentation stage of the automated dataset creation pipeline, pointcloud segmentation was conducted with a modified version of the open-source Depth Clustering ROS node [16], [53]. This segmentation stage output '.pcd' files, each containing a single unlabeled segmented pointcloud for processing in the subsequent stage. Figure 21 depicts Depth Clustering applied to single pointcloud scene, producing four pointcloud segments representing discrete physical objects.



From left to right, segment identities are tree truck, SUV, hazard sign, and car.

Figure 21. Phase 1's Segmentation Step Implemented Using Depth Clustering

3. 3D-2D Correlation

We implement Phase 1's 3D-2D correlation step using the approach outlined in our model overview and applying it with the Logitech c920's specifications. As noted, the c920's horizontal FOV and vertical FOV are approximately 70.42 degrees and 43.30 degrees, respectively. We configure the c920 to collect a full HD frame of 1920x1080 pixels and, applying the formula described in our model outline, calculate a horizontal pixel-per-degree ratio of 27.26 pixels/degree and a vertical pixel-per-degree ratio of

24.94 pixels/degree. Applying the arc tangent and linear translation operations described in our model's 3D-2D correlation step, we show our implementation's capability of mapping pointcloud segments to a synchronized 2D frame. NumPy (a data manipulation library) slicing operations are employed to simply crop the 2D frame to the bounding box defined during the 3D-2D correlation process [54]. OpenCV's basic image processing operations are used to overlay correlated points and bounding boxes on the 2D frame for visualized feedback and debugging [55]. Figure 22 depicts a pointcloud representing a bush being correlated to a bounding box on a 2D image frame.

Figure 22. Implementation of 3D-2D Correlation of 3D Pointcloud Segment to Bounding Box on 2D Frame

4. 2D Classification

We implement Phase 1's 2D classification step using industry-standard, neural network-based image classification tools. TensorFlow and Inception-V3, detailed in Section IV, analyze the 2D image crop produced during 3D-2D correlation and provide a list of plain English labels for the image crop selected from a total of 1000 available class

types. As previously noted, the Inception architecture scale image input to uniform 299x299 image dimensions and we allowed TensorFlow to handle all scaling and warping operations for our image crops. To accelerate Inception's processing of 2D image crops into lists of prospective image labels, we leveraged an Nvidia GTX1050 GPU. The results from Inception-v3's classification processing were rank-ordered based on confidence value and needed to be filtered prior to insertion into our dataset.

5. Confidence-Level Thresholding

The final Phase 1 step implemented during our research was confidence-level thresholding. For the purposes of our research, we selected a default confidence threshold value of 70%, an arbitrary selection. A more thorough evaluation of a potential Phase 1 confidence threshold values could improve future performance by determining an optimal value for a given environment. Our pipeline's implementation of confidence-level thresholding simply discarded any pointcloud segment whose plain English label from Inception-v3 failed to breach 70%. Segments whose labels were greater than or equal to 70% were added to the finished, high-confidence dataset comprised of labeled pointclouds.

6. Phase 1 Steps Not Implemented

Our implementation focused on the automated dataset creation portion of Phase 1 and, as such, made no attempt to implement and evaluate our model's finals steps of Phase 1: segment transformation and neural network training. However, other research has previously shown the validity of these two approaches in producing neural network-based pointcloud classifications models [8].

B. PHASE 1 EXPERIMENT SETUP

We evaluate our Phase 1 implementation's performance on two of the reprocessed dataset collections, referred to as "Neighborhood 1" and "Neighborhood 2," which contain a variety of objects typical of suburban neighborhoods. Figure 23 shows the collection routes driven for the two datasets.

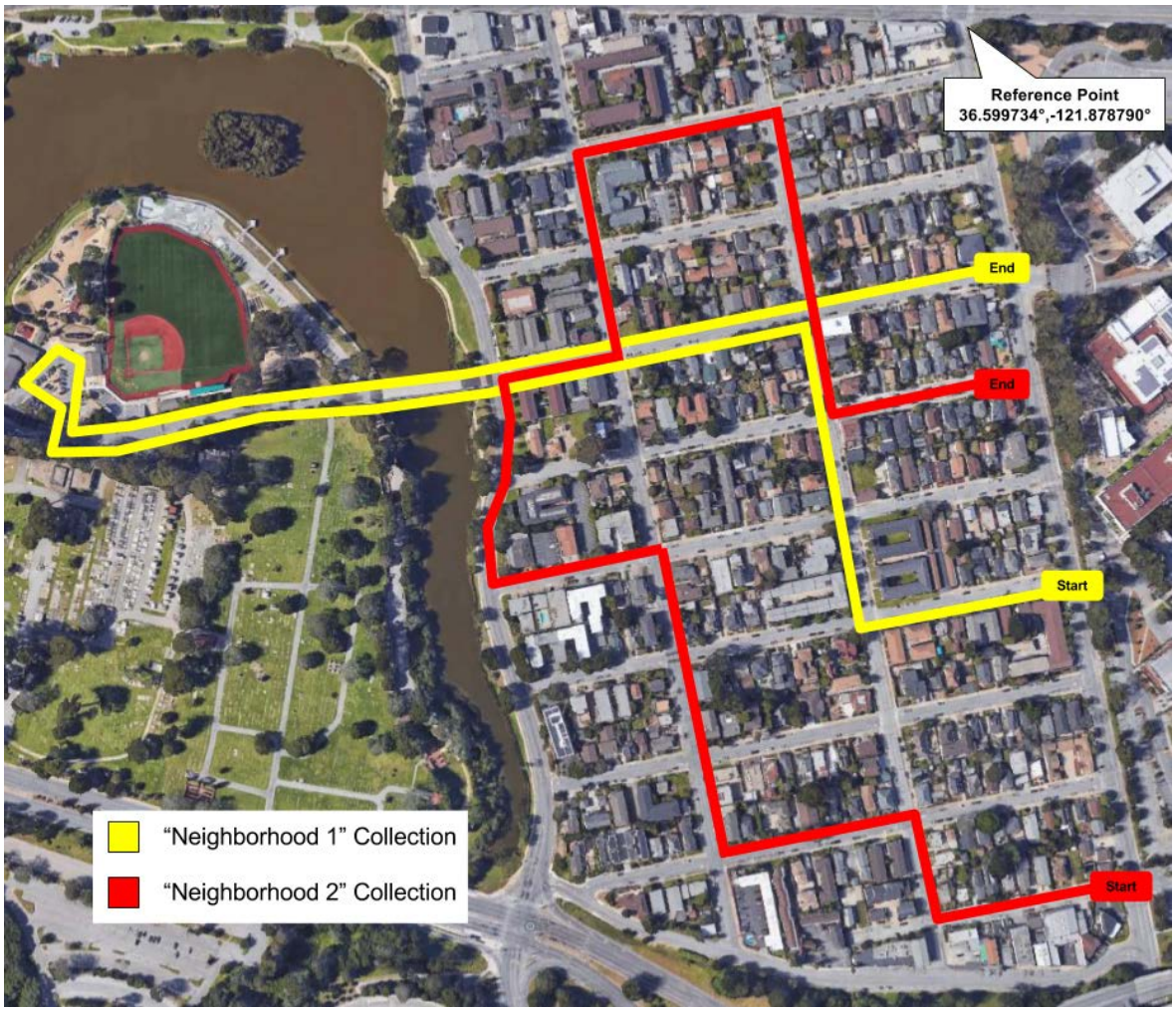


Figure 23. Map View of Routes for "Neighborhood 1" and "Neighborhood 2" Collections

C. PHASE 1 CRITERIA FOR EVALUATION

The criteria to measure the accuracy of the Phase 1 implementation is as follows. First, for each of the labeled pointcloud segments output by the pipeline, the resultant label is manually compared to the original cropped image and assessed for accuracy in labeling. This is necessary to determine whether a classification error is caused by Inception-v3. Second, each pointcloud segment is visualized and manually compared to its corresponding 2D crop. This step is necessary to detect two sources of error: 1) to determine whether a synchronization issue between the LIDAR frame and camera frame caused the 2D image crop to not contain the 3D pointcloud segment and 2) to determine

whether a foreground object in the 2D image crop inappropriately produces a label for a 3D pointcloud segment in the background (occlusion error). Cases that exhibited none of these errors are deemed “correctly” labeled and outputs containing any of the aforementioned errors are deemed “incorrectly” labeled. Of particular note, any four-wheeled vehicle label is considered “correctly” labeled even in cases where the classifier produced an overly specific semantic label for the four-wheeled vehicle (e.g. “beach wagon”). Further, poorly segmented LIDAR frames can cause small artifacts to be improperly included in pointcloud segments. For instance, a segmented pointcloud containing a vehicle might contain a few errant points from an overhanging tree, or from surrounding pavement, that were not properly removed from the segmented pointcloud. These results are deemed acceptable. Further, pointcloud segments containing duplicated objects due to bad laser returns are ignored. For each raw dataset tested, the accuracy of the pipeline is evaluated based on the fraction of segments that qualify as “correctly” labeled relative to the total number of labeled segments produced by the pipeline.

D. PHASE 2 EXPLORATION

As additional research, we developed an implementation of Phase 2’s context discovery methodology. No attempt was made to evaluate the performance of our Phase 2 implementation and we provide our design to shape future research. Our implementation executed the media acquisition step by bulk video scraping from YouTube, object detection through the YOLO9000 neural network, and the creation a customized SQL schema to create a SQL scene context database containing our results. Details of our implementation are as follows.

1. Acquisition of Geotagged Media

Phase 2’s initial step, acquisition of geotagged media, requires the collection of media files recorded in a specific operating environment. Our exploratory research focused on identifying geotagged YouTube videos and used Google’s API to conduct search queries for videos recorded in the desired area, providing a large list of URLs for videos meeting our search criteria. Google’s search API does not offer download capability, however, the open source youtube-dl python module was used to bulk-

download the large list of YouTube video URLs returned by our API search. As such, we demonstrated the ability to script the bulk downloading of geotagged videos assessed to have been recorded in our targeted operating environment. These files represent a large pool of media from our operating environment and hold important information on the spatial relationships between objects in that environment. To uncover these relationships, we process each file through an object detection neural network during the next step in Phase 2.

2. Object Detection

The object detection step of Phase 2 maps out information on the prevalence of specific object types in a specific operating environment, as well as the spatial relationships between these object types. During our exploratory work on Phase 2's object detection step, we employed the YOLO9000 object detection neural network as a fast mechanism to evaluate each frame of every video previously scraped during Phase 2's media acquisition step. Our implementation was shown to process five concurrent videos for object detection and relationship identification on a 16-core/32-thread Xeon CPU with nVidia Quadro M5000 GPU. YOLO9000 provided object detection classifications on frames that contained only a solitary identifiable object and on frames with multiple objects. In the subsequent step, both of these results get archived as an absolute detection count and as weighted relationship identifications, respectively.

3. Populate Scene Context Database

The final step of Phase 2 involved populating our object detection results into a "scene context" database. In our exploratory implementation, we developed a custom SQL schema implementing our model's outlined approach for the YOLO9000 class space. Executing our SQL schema using the MySQL interface, we created a database containing 9418 tables, one for each of YOLO9000's classes. Each table was designed to accommodate up to 9419 column entries, with 9418 representing weighted object relationships identified through object detection and a single column indicating the total number of identifications for each object type. Our implementation's SQL schema also contained a lookup table to translate between the cryptic synset identifiers used by

YOLO9000, such as “n03417042,” to plain English names such as “garbage truck.” Lastly, our implementation maintained an additional table to track the YouTube video tags, a unique portion of a YouTube video’s URL, already processed during context discovery. This table was crucial to avoid repeatedly re-processing videos when building the database over multiple sessions.

Figure 24 shows a more easily visualized Phase 2 implementation based on a smaller 90-class COCO model vice the 9418 classes used by YOLO9000. The left side of the diagram, which depicts the 90 tables created for COCO’s class space, is exploded to reveal a single table corresponding to “parking meter.” This example illustrates the results surfaced for “parking meter” objects while processing geotagged media through the object detection step of Phase 2. Specifically, Figure 24 indicates 60 instances of parking meter were detected in the geotagged media, which we refer to as the absolute detection count. Further, three spatial relationships were identified for objects of type “parking meter.” Specifically, a relatively strong spatial relationship between “parking meter” and “car” was identified, judging by the instance count of 10, and a relatively weaker spatial relationship was uncovered between “parking meter” and “person” and “parking meter” and “truck,” based on their low instance counts.

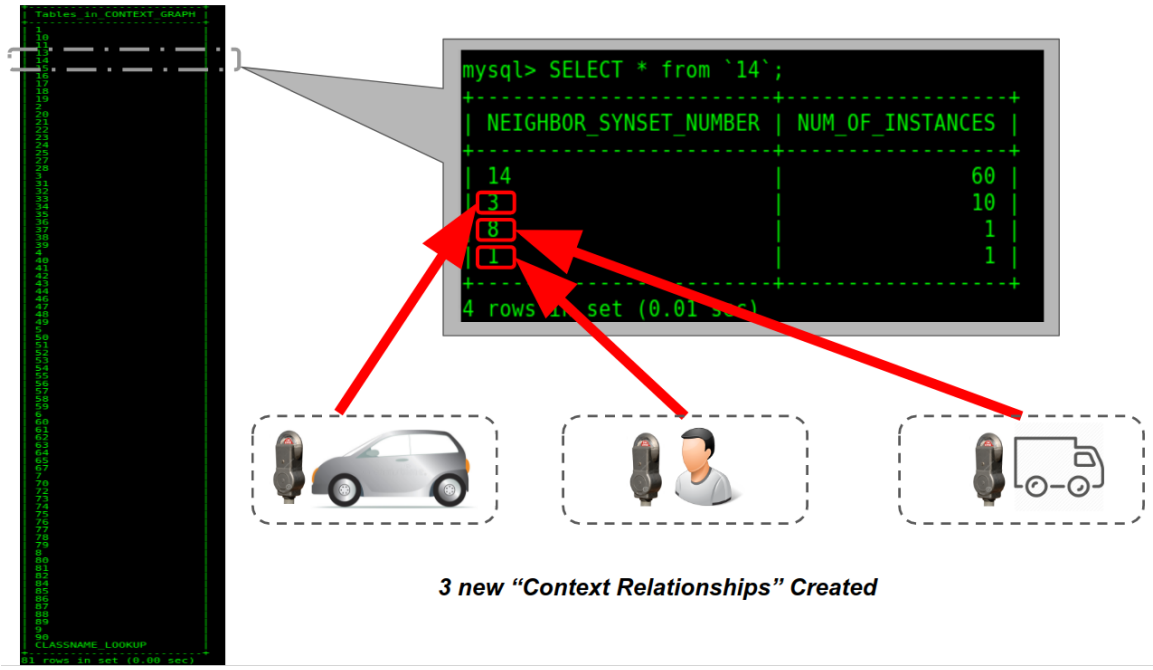


Figure 24. "Scene Context" Database Table for "Parking Meter." Shows Weighted Relationship with "Car," "Person," and "Truck."

We further provide a visualization (Figure 25) of the entirety of spatial relationships mapped out by our Phase 2 implementation on the COCO class space. This particular 90x90 visualization shows each object type's absolute hit count as the dominant diagonal values, with spatial relationships between object types depicted as intersecting values of the graph's 3D grid.

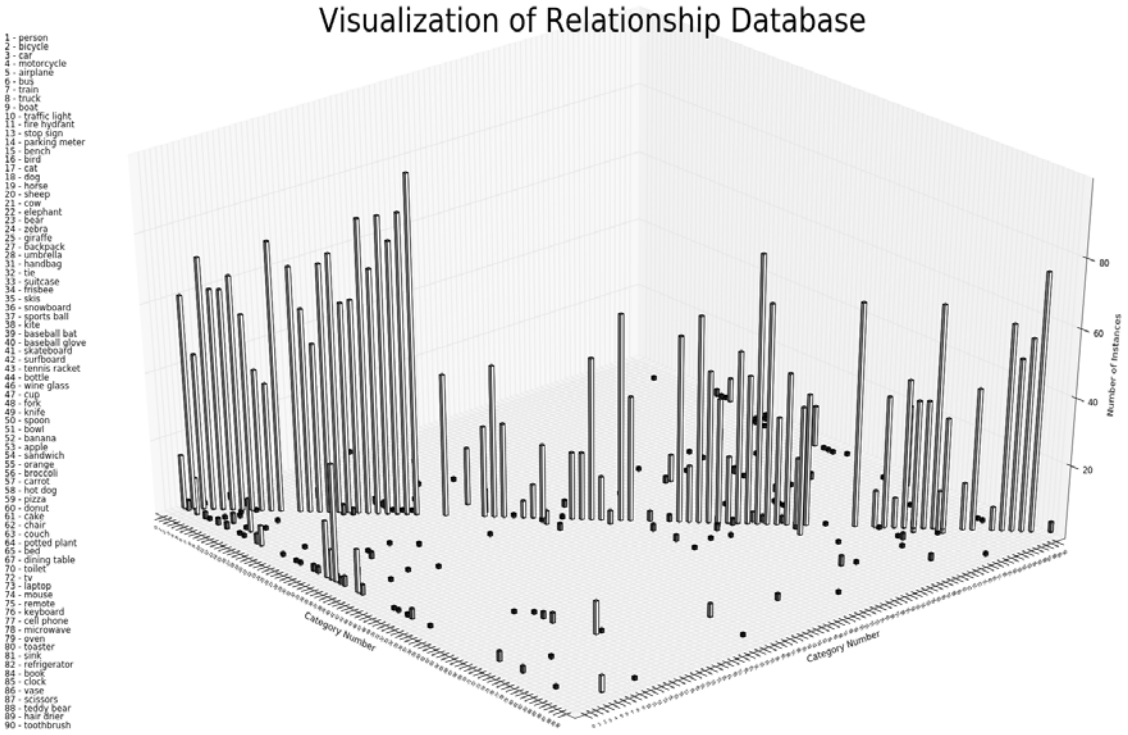


Figure 25. Zoomed-Out Visualization of Entire "Scene Context" Database Containing Relationship Weights between Objects in a Specific Physical Environment

We note that the relationships discovered as part of our Phase 2 implementation required no human intervention, beyond selecting a targeted physical environment. However, significant additional research is in order to assess the accuracy of our Phase 2 implementation.

VI. ERROR ANALYSIS

The model presented in Chapter II encompasses three phases, with each phase providing sources of error affecting the model's performance. The implementation developed for our research focused on Phase 1's automated dataset creation and we provide an analysis of potential errors affecting Phase 1's performance.

A. PHASE 1 ERROR SOURCES

Phase 1, automated dataset creation, suffers from several types of error with varying levels of frequency. Further, errors can percolate and compound throughout multi-stage pipelines, causing unexpected behaviors in the output. As such, we methodically review sources of error for each step of Phase 1.

1. Synchronized Collection Error

Synchronization issues persisted within our Phase 1 implementation, which manifested as poorly mapped bounding boxes during the 3D-2D correlation process. At distances further than a few meters, the impact of sync issues was lessened with respect to the implementation's labeling process. This was due to distant objects moving more slowly across a frame, lessening the impact of synchronization errors on distant objects during the 3D-2D correlation process. The ROS bag file format archived LIDAR and camera messages as they arrived; however, some sync issues persisted. The implementation's output, depicted in Figure 26, showed behavior consistent with LIDAR data being mapped onto "stale" camera frames. Analysis indicated the LIDAR data collected was approximately 170 milliseconds ahead of the camera data, approximately 5 camera frames. The effect of this temporal offset was, in some cases, substantial. As depicted in Figure 26, a cropped bounding box surrounding the blue colored LIDAR segment would almost completely miss the intended lamppost pixels -- making it impossible for TensorFlow to properly determine a label of "lamppost" for such a crop. A calibration solution that manually measured the offset might help resolve the sync issues with a frame skipping solution.

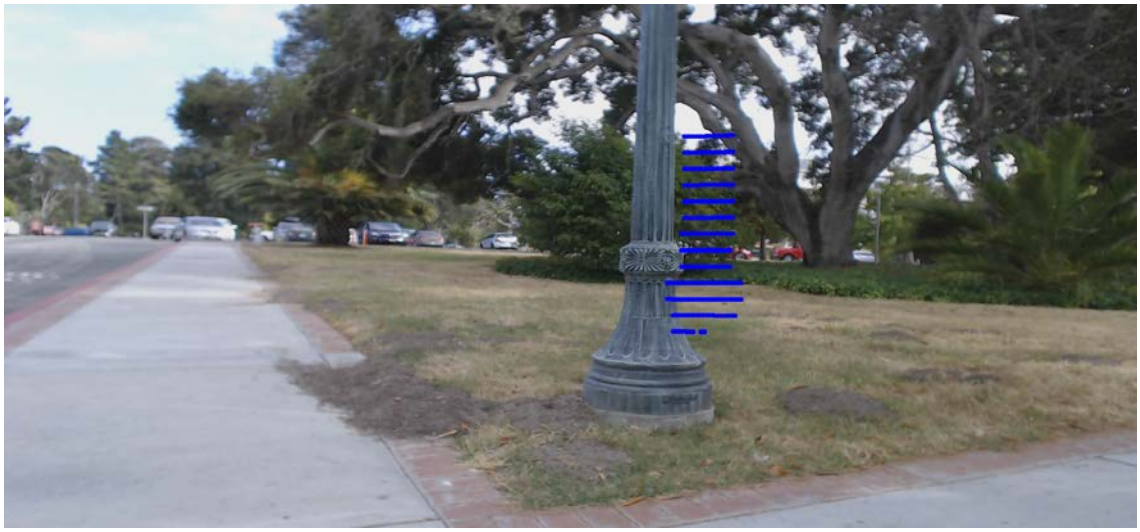


Figure 26. LIDAR Segment Out of Sync with Camera Frame

Other collection errors impacted our model's implementation as well, albeit to a much lesser degree. The raw collection produced by our LIDAR sensor could return a "ghosted" second representation of an object it imaged, giving the impression that the LIDAR inappropriately processed multiple laser returns for the same object. Figure 27 depicts this behavior on a street sign. Notice Figure 27's overhead representation of the segment being processed within the black-background window named "segment." The pointcloud segment collected by the LIDAR clearly shows two distinct street signs whereas the ground-truth representation of the scene provided by the RGB camera shows no second street sign. This source of error was exceptionally rare during development of our Phase 1 implementation, however, future work could increase the accuracy of our implementation by addressing this issue.

Figure 27. LIDAR Collection Error; Ghosted Pointcloud Segment

We assess that this “ghosting” effect is due to mismatch between the LIDAR’s frame of reference and the continuous scanning of the LIDAR hardware. The HDL-32e LIDAR generates a new 360-degree pointcloud scene every 100ms by rotating an array of 32 vertically-aligned lasers around the sensor’s Z-axis. As such, each pointcloud point along a given horizontal plane gets imaged at a different moment. The simultaneous movement of our collection vehicle causes the LIDAR’s frame of reference to change slightly during these 100ms scans. The effect shown in Figure 27 may be the result of these factors. Specifically, we hypothesize that the LIDAR’s vertical array of lasers first image the sign and generate the leftmost representation in Figure 27. Subsequently, the vehicle moves forward and shifts the LIDAR’s frame of reference. Finally, the LIDAR again images the physical sign during the scan, generating Figure 27’s rightmost sign representation.

Another source of potential error in our Phase 1 model stems from errant laser returns in our raw collection. Our implementation exhibited errant, outlier laser returns that were random and limited in density. As such, these errant points never met threshold

to qualify as valid pointcloud segments by the Depth Clustering algorithm and had no impact on the accuracy of our final dataset. Figure 28 depicts one such errant laser return mapped onto a camera frame. Notice the lack of any physical object in the location of the solitary laser return in the center of the frame.

Figure 28. Errant Laser Return Revealed during 3D-2D Correlation

2. Pointcloud Segmentation Error

Our Phase 1 implementation modified a proven, open-source Depth Clustering ROS node to conduct LIDAR scene segmentation. However, some segmentation of pointcloud scenes resulted in segments containing multiple discrete scene objects. Figure 29 shows an example of this behavior while using our implementation's segmentation technique and it represented a source of error in our finalized dataset. As with the case of Figure 29, the bulk of LIDAR points belong to the parked vehicle not the two street signs, however, a 2D crop of the bounding box, when processed by a 2D image classifier, led to inconsistent labels getting produced for segments containing multiple scene objects.

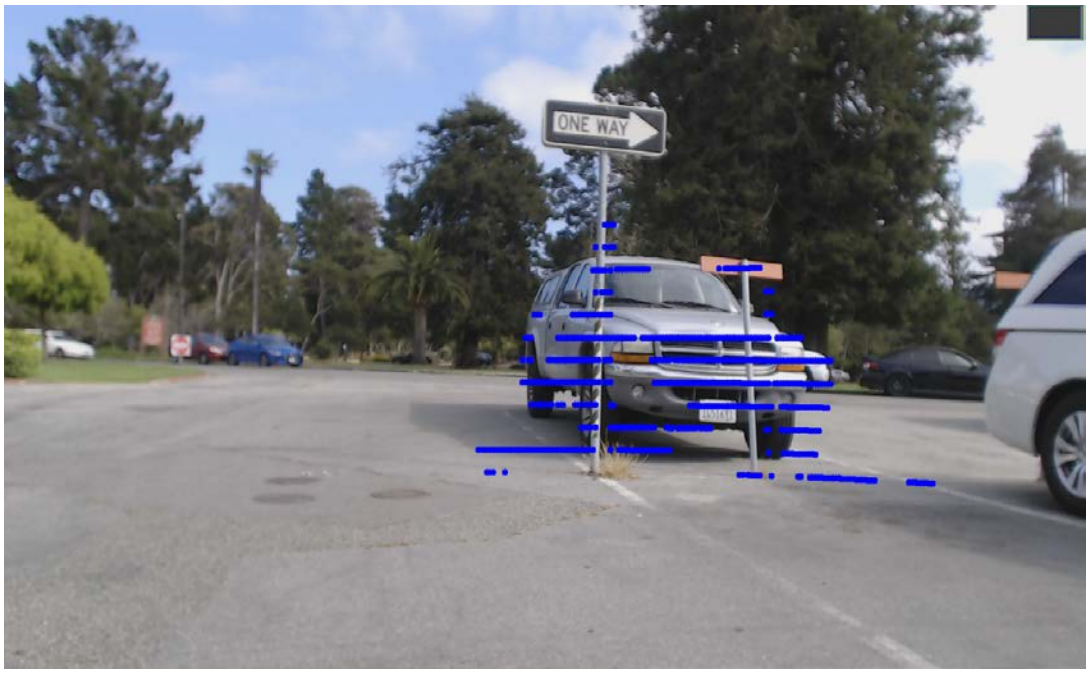


Figure 29. Vehicle and Signs Segmented as a Single Object

3. 3D-2D Correlation Error

The error imposed by 3D-2D correlation on our Phase 1 implementation was subtle and had little impact due to the robustness of our 2D image classifier. These types of errors spawn from not accurately mapping a 3D point onto the 2D image used to create a segment's label. As such, these errors can have the effect of shifting the 2D image crop off the correct location of the physical object within the frame. This is distinct from correlation errors caused by poor synchronization between the LIDAR and camera sensors, which also cause the 2D image crop to be shifted away from the intended object. Figure 30 visualizes the impact of subtle 3D-2D correlation error. In Figure 30, the collection platform is moving towards the center of the frame, between the highlighted rooftop and highlighted pole. We assess Figure 30 exhibits 3D-2D correlation error because a synchronization error would, in this case, map both the rooftop points and pole points to pixels on the outer edge or inner edge of the objects on the 2D frame. In Figure 30 both the pole and rooftop's points are mapped to their left edge, which could only be triggered by a sync issue if the vehicle was turning. We found these offsets to be subtle relative to the error incurred from synchronization and we assess they may be the easiest

source of error to remedy. The integration of a pre-calibrated 360-degree camera, or manually calibrating the cameras from our current implementation, may limit this source of error.

LIDAR points mapping to left of objects located on left and right edge of frame (pole and rooftop) as collection platform drives between them.

Figure 30. Horizontal 3D-2D Correlation Error

4. 2D Classification Error

Our Phase 1 model's reliance on 2D image classification presents a clear source of potential error, as any inaccurately labeled segments have the potential to be inserted into the dataset. Our Phase 1 implementation, which relied on Inception-v3 for 2D image classification, struggled with accurately identifying many small image crops as shown in Figure 31's classification of tree leaves as "bald eagle, albeit with 3% confidence. Low-confidence, incorrect classification results such as seen in Figure 31 have no impact on Phase 1's accuracy because they are easily filtered out by the confidence-level filter.

Figure 31. 2D Image Classification Error

Another source of error in Phase 1's 2D image classification step is occlusion. While processing the collected dataset through our implementation, the labeling of distant objects showed sensitivity to foreground occlusion. Specifically, a properly segmented LIDAR pointcloud might have some portion of its correlated 2D image crop contain a foreground object. This foreground object could dominate the 2D image classification processing of a 2D crop and incorrectly assign the foreground object's label to the pointcloud segment of a background object. Figure 32 provides an example of a pointcloud segment representing a tree receiving a label of "convertible car" due to the presence of a car's hood in the 2D crop.

Figure 32. Occlusion Example. Tree Pointcloud Segment Labeled as Car Due to Foreground of Image Crop.

A more careful scheme for calculating bounding boxes, such as one conducting ray casting on each unlabeled segment for collision testing with other pointcloud segments in the scene, could limit the impact of foreground occlusion on our implementation.

5. Confidence-Level Thresholding Error

Perhaps the most disappointing type of error exhibited by our Phase 1 implementation are the instances of high-confidence, yet incorrect results which make it past the confidence value thresholding filter. While most incorrect 2D image crop classifications are filtered due to low confidence levels, there are periodic instances of small image crops that are incorrectly labeled by Inception-v3. Figure 33 depicts our implementation attempting to label a small LIDAR segment containing a portion of a tree. The small 3D pointcloud segment in turn produces a small 2D image crop, which is labeled as a “bearskin” with a 65% confidence value. While slightly below our proposed

70% threshold, this result illustrates the potential for incorrectly labeled segments to be inserted into the final dataset.

Figure 33. Incorrect, High-Confidence Pointcloud Label of “Bearskin” for Tree

Without more context (e.g., a larger 2D crop of the object and surroundings), it may not be possible for a 2D image classifier to label these types of crops. However, enlarging the 2D crops with a buffer of additional pixels around the crop’s perimeter could increase occlusion errors. Further, improvements to 2D image classifiers’ capabilities with respect to labeling small image fragments, or employing a more sophisticated scheme for pre-processing small image crops prior to classification, could significantly improve our implementation’s performance in this area.

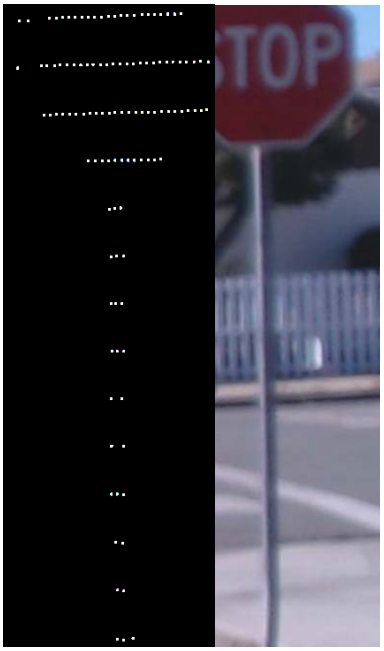
THIS PAGE INTENTIONALLY LEFT BLANK

VII. RESULTS

Evaluation of our implementation of Phase 1's automated dataset creation found it capable of producing valid, labeled pointcloud segments with varying levels of accuracy. We first assess the suitability of our Phase 1 implementation's functionality and then measure the accuracy of its labeled dataset output.

A. PHASE 1 FUNCTIONALITY

Our Phase 1 implementation processes synchronized LIDAR and camera input and, in many cases, outputs a properly labeled LIDAR pointcloud segment alongside the accompanying 2D image crop. Figure 34 depicts the properly functioning pipeline operating on the dataset collected - the automated pipeline applying a correct "street sign" label to a corresponding LIDAR pointcloud segment.



LIDAR Segment Properly Labeled as "Street Sign" based on the 2D Image Classification of the Corresponding Image Crop

Figure 34. Example of Phase 1 Implementation Output

B. PHASE 1 PERFORMANCE

The dataset “Neighborhood 1” was processed through the automated pipeline and created a total of 41 labeled pointcloud segments from a collection duration of 346 seconds. Based on the following criteria, we judge the pipeline to have produced 31 correctly labeled pointcloud segments and ten incorrectly labeled pointcloud segments, representing a 75.6% accuracy (Figure 35). A detailed analysis of the “Neighborhood 1” results is provided in the Appendix, Section H.

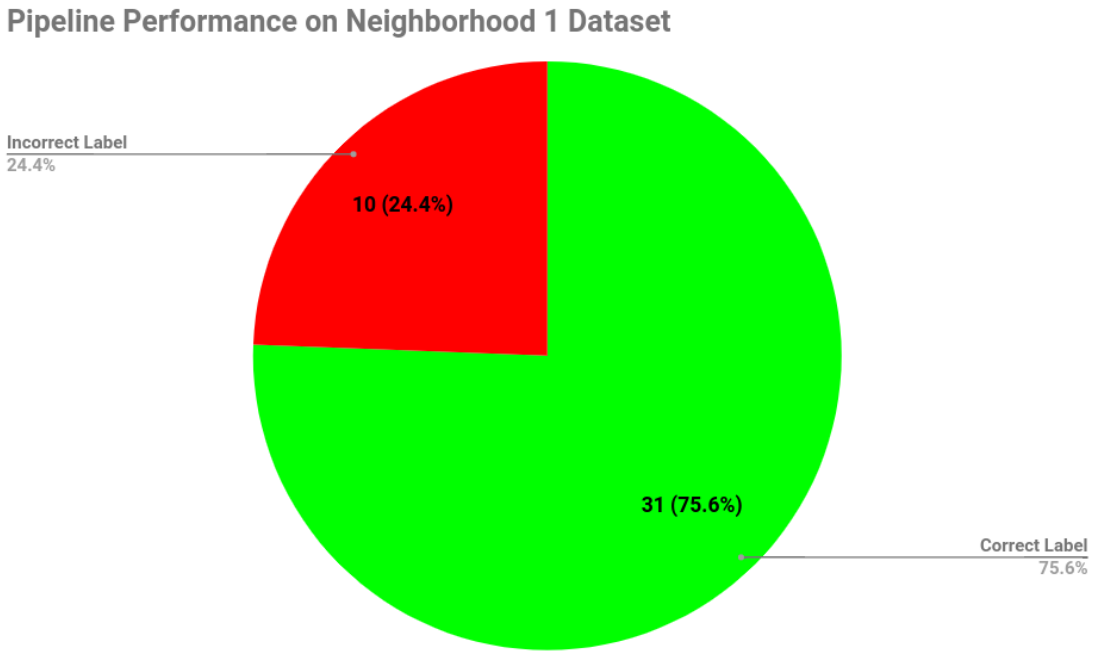


Figure 35. Pipeline Performance on “Neighborhood 1” Dataset in Producing Correctly Labeled Pointcloud Segments

1. Sources of Pipeline Error

The ten incorrectly labeled segments suffered from incorrect image classification, synchronization issues, and occlusion errors. The two incorrect image classification results were not necessarily due to poor accuracy in Inception-v3’s classification but, instead, were due to extremely small image crops being fed into Inception-v3. For context, our implementation neglected to do any filtering of small image crops and,

therefore, allowed Inception-v3 to automatically upscale and warp aspect ratios to its 299x299 standardized input dimensions. Small image crops, a result of small pointcloud segments, can occur in the pipeline when an imperfect segment is created or, more frequently, when an object slowly moves into the LIDAR's field of view, causing it to capture only a sliver of the object. The latter circumstance was the likely cause in both of these instances. In one case, a small slice of a tree produced a very small, dark, and splotched image crop that very closely resembled a bearskin, leading to a high confidence, yet incorrect semantic label. This behavior could likely be mitigated by ignoring segments on the periphery of the LIDAR scene or, possibly, by imposing a minimum pointcloud size. A single output of the 41 results was incorrect due to synchronization issues. This case, a thin pole, was illustrative of the narrow circumstances where synchronization could trigger an incorrect result not being filtered from the results. Inception-v3 is capable of classifying objects only partially contained within an image crop, adding significant robustness when encountering unsynchronized LIDAR and camera data. However, LIDAR segments containing very thin objects have the potential to be entirely out of frame during the cropping process, ultimately feeding an image crop to Inception-v3 that doesn't contain the intended object whatsoever. Reviewing this pole's long and thin image crop, the intent to capture a pole was obvious and this incorrect semantic label was clearly a victim of unsynchronized data streams.

The primary source of error was occlusion error, with six of the 41 results exhibiting this trait. In all six cases, the cropped frame contained multiple objects and Inception-v3 correctly classified an unintended object, one which did not match the corresponding segmented pointcloud. As the data was collected in a suburban environment, it was common for trees or vehicles to enter the foreground of cropped image, raising the potential for Inception-v3 to classify that object instead. Interestingly, in some cases, the reverse occurred as well, with Inception-v3 correctly classifying a distant background object instead of the intended foreground object. To mitigate this behavior, a more exacting approach to creating image crops would be required to avoid including unintended background or foreground objects.

Lastly, one result received a correct label, but was deemed incorrect due to the pointcloud segment being small and difficult to definitively identify as the same object in the crop.

The pipeline was subsequently run on the "Neighborhood 2" dataset, which measured 317 seconds in length, and achieved greater accuracy (see Figure 36). This second test produced a total of 35 labeled segments with 31 evaluated as "correct" and four as "incorrect." A detailed analysis of the "Neighborhood 2" results is provided in the Appendix, Section I.

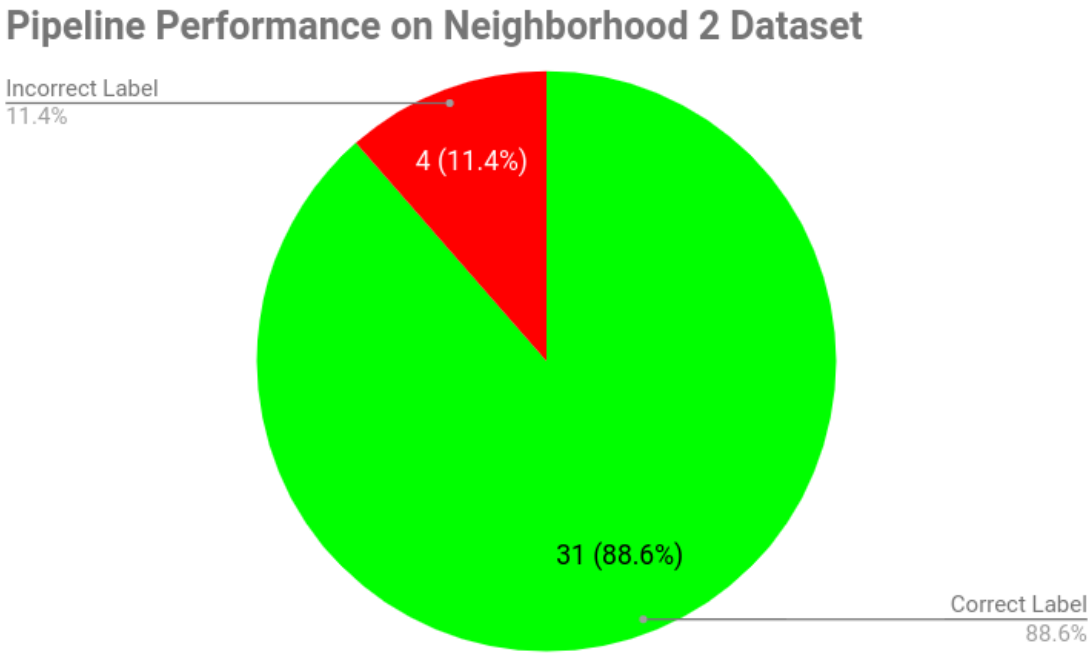


Figure 36. Pipeline Performance on "Neighborhood 2" Dataset in Producing Correctly Labeled Pointcloud Segments

The pipeline performed better on this dataset due to the higher incidence of vehicles compared to Neighborhood 1, as the pipeline is able to consistently produce vehicle labels. Two of the four incorrect labels suffered from synchronization errors due to the image crop not containing the appropriate object, and the remaining two incorrect

labels stemmed from poor segmentation and classification of a background object instead of the intended foreground object.

The aggregate performance of the pipeline, when combining the results of both datasets Neighborhood 1 and Neighborhood 2, reached the level of 81.6% accuracy (see Figure 37). With the two collection's combined duration of 663 seconds, the pipeline created a correctly labeled segment every 10.7 seconds and an incorrectly labeled segment every 47.4 seconds. The diversity of the resultant labeled dataset was extremely limited due to time-imposed practical constraints. As previously noted, the area processed by our pipeline was directly in front of the collection vehicle and the collection was culled to 10 meters of maximum depth with a 70-degree horizontal field of view. This culling largely limited the ground-truth diversity of object types available for classification to a mixture of street objects, such as parked vehicles, street signs, motorcycles, and landscaping. Most of the physical space processed by the pipeline contained open road. This produced a dataset heavily biased towards our limited collection environment, with Neighborhood 1's results comprised of over 50% vehicles. Moreover, 100% of Neighborhood 2's correctly labeled results were from vehicles. Our implementation requires testing in different operating environments to assess its effectiveness in producing diverse datasets.

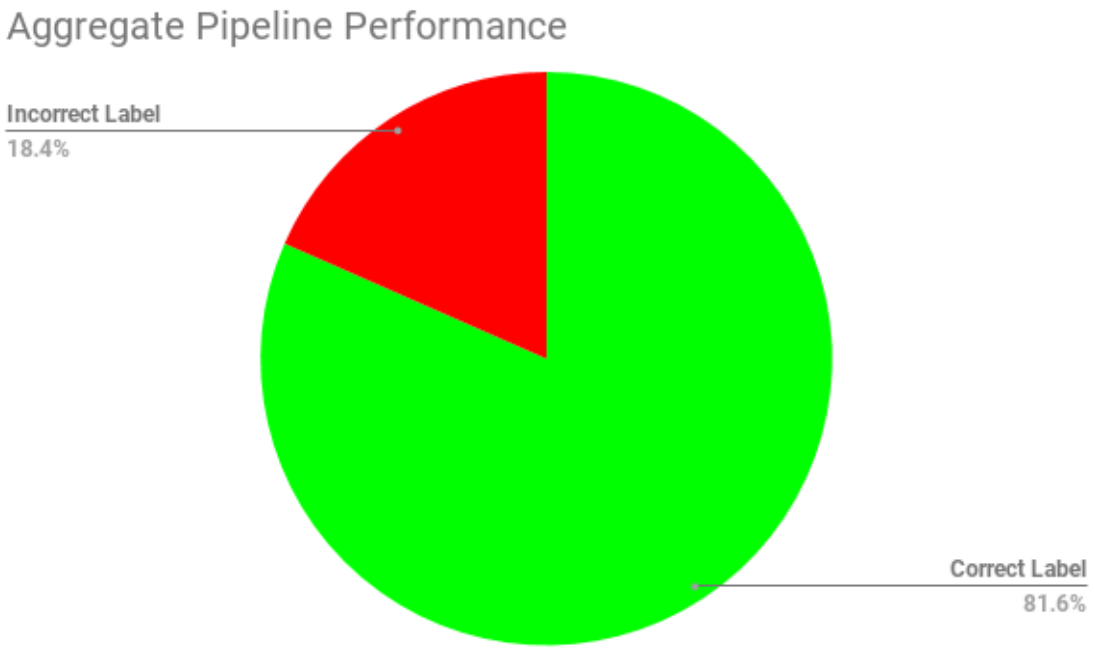


Figure 37. Aggregate Performance of Automated Dataset Creation Pipeline

2. Comparison to Human Performance

Comparison to human performance in hand annotation is exceedingly difficult due to varying levels of human performance and variations in human background knowledge of the classification ontology (e.g. 1000 classes in ImageNet) [1]. Most relevant, even when measuring human performance on a Top-5 evaluation basis (e.g., any the top 5 “guesses” are considered when judging a correct classification), human annotation error rates on samples from the ImageNet dataset ranged from 5.1% - 12% [1], with less rigorous data showing untrained annotators achieving a meager 15% Top-5 error rate [1], [7]. Top-1 error rates for human annotation are likely significantly higher than the 5.1 - 15% error rate achieved with the Top-5 evaluation method. As our pipeline relies on Top-1 labels and is not able to leverage the menu of labels provided by Top-5 results, we characterize the upper bound on Top-1 accuracy for human annotation at approximately 85 - 95% for the 1000 class ImageNet dataset. For comparison, our pipeline, which relies on the Top-1 result for labeling LIDAR data, achieved a combined accuracy of 81.5%, with 62 correctly labeled pointclouds and 14 incorrectly labeled

pointclouds. Conservatively, this 81.5% accuracy of LIDAR data is comparable to the 85 - 95% Top-1 accuracy of human annotation of 2D images; however, it is our assessment that, in practice, human performance on hand labeling low resolution LIDAR segments could be considerably lower than 85 - 95% accurate without the help of additional annotation tools or significant investment of time in training human operators to recognize objects within low resolution LIDAR pointclouds. Further, we assess that human validation of class labels assigned by the pipeline is faster than human-based annotation from scratch, particularly when assigning labels from a large list of class types such as ImageNet's 1000 class types.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. CONCLUSIONS

Our research sought to evaluate whether state-of-the-art 2D image classification neural networks can create labeled training data for pointcloud-based neural network classifiers. In pursuit of our research question, we created an automated pipeline for creating labeled LIDAR data that combined the automated image classification and pointcloud segmentation capabilities of TensorFlow's Inception-v3 and Depth Clustering, respectively. We provide our conclusions regarding the effectiveness of this approach, as well as an overview of potential areas for future work in this field.

A. AUTOMATED DATASET CREATION CONCLUSIONS

Our initial results were promising, but require additional research to show the advantages of our model over established dataset creation techniques. Our pipeline only showed 81.6% accuracy of labeled data outputs and our pipeline's throughput was limited, averaging only one correctly labeled segment every 10.7 seconds of raw data. Moreover, our implementation's ability to produce diverse datasets is unknown, as we produced two datasets containing very few object types. Lastly, compared to established pointcloud annotation techniques, our model has the added burden of a second hardware sensor (RGB sensor) and its labeling capabilities are constrained to the class space of existing 2D image classifiers, whereas human beings could feasibly label a much larger range of objects.

The pipeline's output could be useful as a "first draft" of a finalized training dataset, particularly for problem sets with extremely large quantities of raw data that would otherwise be untenable to hand annotate. Specifically, the pipeline is scalable and could be parallelized to process large datasets that are currently restricted to human crowdsourcing techniques. A "first draft" approach might leverage the pipeline to provide an initial dataset with approximately 80% accuracy, leaving a human annotator to simply validate the results. Additional research is required to determine whether such an approach would offer improvements over existing hand annotation approaches.

Given the wide range of remaining issues with the creation of training datasets for neural networks, which are likely to hold back the application of neural networks to a variety of problem sets, it is paramount to continue developing efficient tool-based approaches, such as the pipeline proposed here, and ensure that existing datasets are fully utilized.

B. FUTURE WORK

The accuracy of the pipeline's annotations leaves significant room for improvement. In addition to improvements to the model's implementation, we posit that future improvements in industry-standard 2D image classification performance could have a direct benefit to our model's accuracy performance. In closing, given the difficulties associated with creating these labeled training datasets, we propose establishing a centralized database to retain and maximum the utility of existing training datasets.

1. Dataset Database

Moving forward, as neural networks become integral to ever-larger numbers systems, an increasing amount of labeled training data will be required, likely created at significant expense, in order to train these systems. To maximize the benefit of the substantial cost of creating this labeled training data, the machine learning community could further encourage the use of community-wide repositories for labeled training datasets. We assess this could help offset the one-time cost of dataset creation by amortizing their use across multiple applications. While some labeled datasets may be created for niche applications with limited options for reuse, others will likely contain more common objects, such as vehicles, and present clear opportunities for reuse in other applications.

APPENDIX

A. TANDEM LIDAR AND CAMERA MOUNT TOP PLATE CAD DRAWINGS

https://gitlab.com/chambana/LIDAR_Object_Recognition/blob/master/Tandem_LIDAR_and_RGB_Mount/CAD/LIDAR_and_4_camera_tandem_mount.skp

B. TANDEM LIDAR AND CAMERA MOUNT MOBILE COLLECTION BOX CAD DRAWINGS

https://gitlab.com/chambana/LIDAR_Object_Recognition/blob/master/Tandem_LIDAR_and_RGB_Mount/CAD/mobile_collection_box.skp

C. TANDEM LIDAR AND CAMERA MOUNT TOP PLATE STEREOLITHOGRAPHY FILES

https://gitlab.com/chambana/LIDAR_Object_Recognition/blob/master/Tandem_LIDAR_and_RGB_Mount/CAD/LIDAR_and_4_camera_tandem_mount.stl

D. TANDEM LIDAR AND CAMERA MOUNT MOBILE COLLECTION STEREOLITHOGRAPHY FILES

https://gitlab.com/chambana/LIDAR_Object_Recognition/blob/master/Tandem_LIDAR_and_RGB_Mount/CAD/mobile_collection_box.stl

E. NEIGHBORHOOD 1 DATASET

The Neighborhood 1 dataset collected as part of this thesis is available at https://wiki.nps.edu/x/RwAxNw

F. NEIGHBORHOOD 2 DATASET

The Neighborhood 2 dataset collected as part of this thesis is available at https://wiki.nps.edu/x/RwAxNw

G. ADDITIONAL DATASETS

The following additional datasets were collected as part of this thesis:

NPS Tandem LIDAR Camera Data (San Clemente Island, CA)

NPS Tandem LIDAR Camera Data (Campus)

They are available at https://wiki.nps.edu/x/RwAxNw

H. SEGMENT ANALYSIS OF NEIGHBORHOOD 1 DATASET

filename suffix	>70% label	2D image contents	3D segment contents	2D matches 3D?	Accurately labeled pointcloud?	comments
8312	beach wagon	SUV	SUV	yes	yes	segment has tree artifact over car
8715	bear skin	black and gray pixels, probably foliage	probably foliage, very sparse	yes, but not a bear skin	no	really small crop
8684	convertible	car door	car door	yes	yes	
20740	crash helmet	long empty image, background	likely pole or tree trunk	no	no	sync error made the 2D crop miss the object
16088	fire engine	back corner of vehicle	likely back corner of vehicle	yes	yes	
10747	minivan	foreground car background tree	tree	no	no	occlusion. good 2D classification of foreground object
15926	minivan	back corner of vehicle	back corner of vehicle	yes	yes	
15930	minivan	back corner of vehicle	back corner of vehicle	yes	yes	
15933	minivan	back corner of vehicle	back corner of vehicle	yes	yes	
15937	minivan	back corner of vehicle	back corner of vehicle	yes	yes	
15940	minivan	back corner of vehicle	back corner of vehicle	yes	yes	
15946	minivan	back corner of vehicle	back corner of vehicle	yes	yes	
15949	minivan	back corner of vehicle	back corner of vehicle	yes	yes	
16137	minivan	back of vehicle	back of vehicle	yes	yes	
20546	minivan	back of minivan	back of minivan	yes	yes	
22195	minivan	front corner of vehicle	front corner of vehicle	yes	yes	
22202	minivan	front corner of vehicle	front corner of vehicle	yes	yes	

22388	minivan	back corner of vehicle	back corner of vehicle	yes	yes	
22680	minivan	side doors of vehicle	unknown	unknown	no	possible match, but small segment is difficult to discern
11082	mobile home	foreground tree, home background	tree	no	no	occlusion. good 2D classification of foreground object
11094	mobile home	foreground tree, home background	tree	no	no	occlusion. good 2D classification of foreground object
25162	motor scooter	motor scooter	motor scooter	yes	yes	
25174	motor scooter	motor scooter	motor scooter	yes	yes	small artifacts
25184	motor scooter	back half of motor scooter	likely back half of motor scooter	yes	yes	very sparse segment
25209	motor scooter	middle section of motor scooter	likely middle section of motor scooter	yes (likely)	yes	very sparse segment
10493	picket fence	foreground tree, fence background	tree	yes	no	occlusion. good 2D classification of foreground object
22455	Polaroid camera	lower front bumper of car	lower front bumper of car	yes	no	very small crop
8797	police van	back half of pickup truck	back half of pickup truck	yes	yes	
8822	police van	back half of pickup truck	back half of pickup truck	yes	yes	
12850	racer	foreground stop sign, background car	stop sign	yes	no	occlusion. good 2D classification of background object
12855	racer	foreground stop sign, background car	stop sign	yes	no	occlusion. good 2D classification of background object
9534	street sign	street sign	street sign	yes	yes	
9547	street sign	street sign	street sign	yes	yes	
16787	street sign	street sign	street sign	yes	yes	
16788	street sign	street sign	street sign	yes	yes	

16792	street sign	street sign	street sign	yes	yes	
16795	street sign	street sign	street sign	yes	yes	foliage artifact
16798	street sign	street sign	street sign	yes	yes	
16801	street sign	street sign	street sign	yes	yes	double LIDAR return artifact
16809	street sign	street sign	street sign	yes	yes	
16812	street sign	street sign	street sign	yes	yes	

I. SEGMENT ANALYSIS OF NEIGHBORHOOD 2 DATASET

filename suffix	**>70% label**	**2D image contents**	**3D segment contents**	**2D matches 3D?**	**Accurately labeled pointcloud?**	comments
10730	cab	front half of vehicle	front half of vehicle	yes	yes	
2127	limousine	side of vehicle with person entering	side of vehicle with person entering	yes	yes	
2135	limousine	side of vehicle with person entering	side of vehicle with person entering	yes	yes	
8864	limousine	back of vehicle, person opening back hatch	back of vehicle, person opening back hatch	yes	yes	
8882	limousine	back of vehicle with hatch open	back of vehicle with hatch open	yes	yes	
8889	limousine	back of vehicle with hatch open	back of vehicle with hatch open	yes	yes	
8896	limousine	back of vehicle with hatch open	back of vehicle with hatch open	yes	yes	
15483	limousine	front corner of vehicle	front corner of vehicle	yes	yes	
4133	microwave	long empty crop	tree trunk	no	no	sync error
4137	microwave	long empty crop	tree trunk	no	no	sync error
2809	minivan	back of	back of	yes	yes	

		vehicle	vehicle			
2830	minivan	back of vehicle	back of vehicle	yes	yes	
3881	minivan	back of vehicle	back of vehicle	yes	yes	
3889	minivan	back of vehicle	back of vehicle	yes	yes	
5338	minivan	front corner of vehicle	front corner of vehicle	yes	yes	small ground artifact
5348	minivan	most of vehicle	most of vehicle	yes	yes	
5388	minivan	side of vehicle	side of vehicle	yes	yes	poor segment
5700	minivan	back of vehicle	back of vehicle	yes	yes	
5715	minivan	back of vehicle	back of vehicle	yes	yes	
5719	minivan	back of vehicle	back of vehicle	yes	yes	
14441	minivan	back of vehicle	back of vehicle	yes	yes	
14449	minivan	back of vehicle	back of vehicle	yes	yes	
14824	minivan	back of vehicle	back of vehicle	yes	yes	
14828	minivan	back of vehicle	back of vehicle	yes	yes	
14835	minivan	back of vehicle	back of vehicle	yes	yes	
14837	minivan	back of vehicle	back of vehicle	yes	yes	
19628	minivan	front corner of vehicle	unknown	yes	no	possibly fine, but very poor segment
20452	minivan	back of vehicle	back of vehicle	yes	yes	
3184	mobile home	foreground fence and tree, background home	fence and tree	yes	no	background object got classified
1190	pickup	back of vehicle	back of vehicle	yes	yes	
1201	pickup	back of vehicle	back of vehicle	yes	yes	
1214	pickup	back of vehicle	back of vehicle	yes	yes	
1217	pickup	back of vehicle	back of vehicle	yes	yes	

19614	school bus	back of vehicle	back of vehicle	yes	yes	marginally acceptable. Very small crop and small segment
3505	sports car	front of vehicle	front of vehicle	yes	yes	very small crop and segment

LIST OF REFERENCES

[1] O. Russakovsky et al., “ImageNet large scale visual recognition challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015. [Online]. Available: https://arxiv.org/pdf/1409.0575.pdf

[2] S. Russell, “DARPA Grand Challenge winner: Stanley the Robot!,” *Popular Mechanics*, Jan. 15, 2006. [Online]. Available: http://www.popularmechanics.com/technology/engineering/robots/2169012

[3] O. Cameron, “An introduction to LIDAR: The key self-driving car sensor,” Voyage, May 9, 2017. [Online]. Available: https://news.voyage.auto/an-introduction-to-lidar-the-key-self-driving-car-sensor-a7e405590cff

[4] J. Hu et al. “Squeeze-and-excitation networks,” presented at CVPR 2017, Hawaii Convention Center, Honolulu, HI, USA, Jul. 26, 2017. [Online]. Available: http://image-net.org/challenges/talks_2017/SENet.pdf

[5] C. Szegedy et al., “Inception-v4, Inception-ResNet and the impact of residual connections on learning,” in *AAAI*, 2017. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/download/14806/14311

[6] A. Canziani et al., “An analysis of deep neural network models for practical applications”, arXiv, Apr. 14, 2017. [Online]. Available: https://arxiv.org/pdf/1605.07678.pdf

[7] A. Karpathy. “What I learned from competing against a ConvNet on ImageNet,” GitHub, Sep. 2 2014. [Online]. Available: https://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

[8] D. Maturana and S. Scherer, “VoxNet: A 3D convolutional neural network for real-time object recognition,” in *2015 IEEE/RSJ Int. Conf. on Intelligent Robots and Syst. (IROS)*, 2015. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7353481/

[9] I. Kadar and O. Ben-Shahar, “SceneNet: A perceptual ontology for scene understanding,” presented at ECCV Workshops, Ben Gurion Univ. of Negev, Beer Sheba, Israel. Sep. 6, 2014. [Online]. Available: https://www.cs.bgu.ac.il/~ilankad/pdfs/2014-Kadar_and_Ben_Shahar-SceneNet_A_Perceptual_Ontology_for_Scene_Understanding.pdf

[10] “Computer vision tasks,” class notes for CS231n: Convolutional Neural Networks for Visual Recognition, Dept. of Comp. Sci., Stanford University, Palo Alto, CA, USA, winter 2015. [Online]. Available: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

[11] W. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The Bulletin of Mathematical Biophysics* vol. 5, no.4, pp. 115-133. 1943

[12] A. Krizhevsky, “Imagenet classification with deep convolutional neural networks,” in *Neural Information Processing Systems Conference*. 2012. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[13] “Results of ILSVRC2014,” ImageNet. Accessed Nov. 1, 2017. [Online]. Available: http://www.image-net.org/challenges/LSVRC/2014/results#det

[14] “Image recognition,” TensorFlow, Nov. 2, 2017. [Online]. Available: https://www.tensorflow.org/tutorials/image_recognition

[15] R. Rusu, “Semantic 3d object maps for everyday manipulation in human living environments,” Ph.D. dissertation, Dept. of Comp. Sci., Technical Univ. of München, Munich, Germany, 2010. [Online]. Available: http://mediatum.ub.tum.de/doc/800632/document.pdf

[16] “Depth clustering,” University of Bonn. Accessed Oct. 30, 2017. [Online]. Available: https://github.com/Photogrammetry-Robotics-Bonn/depth_clustering

[17] T. Hackel et al. “Semantic3D. net: A new large-scale point cloud classification benchmark,” in *ISPRS Annals of the Photogrammetry, Remote Sensing, and Spatial Information Sciences,* 2017. [Online]. Available: https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/IV-1-W1/91/2017/isprs-annals-IV-1-W1-91-2017.pdf

[18] Handprinted Forms and Characters Database. NIST Special Database 19. Accessed Nov. 1, 2017. [Online]. Available: https://www.nist.gov/srd/nist-special-database-19.

[19] R. Fisher, Iris Data Set, 1936. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Iris

[20] A. Casalboni et al., “Amazon Mechanical Turk: Help for building your machine learning datasets,” Cloud Academy Blog, Oct. 23, 2015. [Online]. Available: https://cloudacademy.com/blog/machine-learning-datasets-mechanical-turk/

[21] P. Ipeirotis, F. Provost, and J. Wang, “Quality management on Amazon Mechanical Turk,” in *Proc. of the ACM SIGKDD workshop on human computation,* 2010. [Online]. Available: https://dl.acm.org/citation.cfm?id=1837906

[22] T. Lin et al., “Microsoft COCO: Common Objects in Context,” in *Eur. Conf. on Comput. Vision,* 2014. [Online]. Available: https://arxiv.org/pdf/1405.0312.pdf

[23] “About ImageNet,” ImageNet. Accessed Oct. 28, 2017. [Online]. Available: http://image-net.org/about-stats

[24] J. Deng et al., “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conf. on Comput. Vision and Pattern Recognition*, 2009. [Online]. Available: http://www.image-net.org/papers/imagenet_cvpr09.pdf

[25] “VOC2011 annotation guidelines,” Pascal VOC Challenge. Accessed Oct. 28, 2017. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/guidelines.html

[26] K. Lai et al., RGB-D (Kinect) Object Dataset, 2012. [Online]. Available: http://rgbd-dataset.cs.washington.edu/

[27] A. Singh et al., “Bigbird: A large-scale 3d database of object instances,” in *2014 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014. [Online]. Available: http://people.eecs.berkeley.edu/~pabbeel/papers/2014-ICRA-BigBIRD.pdf

[28] S. Choi et al., A Large Dataset of Object Scans, 2016. [Online]. Available: https://arxiv.org/abs/1602.02481.pdf

[29] M. De Deuge et al., “Unsupervised feature learning for classification of outdoor 3d scans,” in *Australasian Conf. on Robotics and Automation*, 2013. [Online]. Available: http://www.araa.asn.au/acra/acra2013/papers/pap133s1-file1.pdf

[30] A. Boyko, “Efficient interfaces for accurate annotation of 3D point clouds,” Ph.D dissertation, Dept. of Comp. Sci., Princeton Univ., Princeton, NJ, USA, 2015. [Online]. Available: http://gfx.cs.princeton.edu/pubs/_2015_EIF/index.php

[31] S. Charrington, “Training data for autonomous vehicles with Daryn Nakhuda of MightyAI,” *This Week in Machine Learning & AI*, podcast, Oct. 23, 2017. [Online]. Available: https://twimlai.com/twiml-talk-057-training-data-autonomous-vehicles-daryn-nakhuda/

[32] B. Russell et al., “LabelMe: A database and web-based tool for image annotation,” *Int. Journal of Comput. Vision*, vol. 77, no. 1, pp. 157–173, May 2008. [Online]. Available: http://www.faculty.idc.ac.il/arik/seminar2010/papers/Databases/LabelMe.pdf

[33] A. Handa et al., “Understanding real world indoor scenes with synthetic data,” in *Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognition*, 2016. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Handa_Understanding_Real_World_CVPR_2016_paper.pdf

[34] D. Prokhorov, “A convolutional learning system for object classification in 3-D LIDAR data,” *IEEE Trans. on Neural Networks,* vol. 21, no. 5, pp.858-863, May 2010. [Online]. doi:10.1109/TNN.2010.2044802

[35] D. Maturana and S. Scherer, “3d convolutional neural networks for landing zone detection from lidar,” in *2015 IEEE Int. Conf. on Robotics and Automation (ICRA), IEEE,* 2015. [Online]. Available: http://dimatura.net/publications/3dcnn_lz_maturana_scherer_icra15.pdf

[36] “ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2017 overview,” ImageNet. Accessed Nov. 1, 2017. [Online]. Available: http://image-net.org/challenges/talks_2017/ILSVRC2017_overview.pdf

[37] M. Caudill, “Neural networks primer, part I,” *AI Expert,* vol. 2, no. 12, pp. 46–52, Dec. 1987.

[38] M. Kukacka, “Overview of deep neural networks,” in *WDS ‘12 Proc. of Contributed Papers,* 2012. [Online]. Available: https://www.mff.cuni.cz/veda/konference/wds/proc/pdf12/WDS12_117_i1_Kukacka.pdf

[39] C. Szegedy et al., “Going deeper with convolutions,” in *Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognition,* 2015. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR_paper.pdf

[40] “A 2-layer neural network,” class notes for CS231n: Convolutional Neural Networks for Visual Recognition, Dept. of Comp. Sci., Stanford University, Palo Alto, CA, USA, spring 2017. [Online]. Available: http://cs231n.github.io/neural-networks-1/

[41] M. Abadi et al., “TensorFlow: Large-scale machine learning on heterogeneous distributed systems,” in *Proc. of the 12th USENIX Symp. on Operating Syst. Des.,* 2016. [Online]. Available: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf

[42] M. Li, “Ranking Popular Deep Learning Libraries for Data Science,” The Data Incubator, Oct. 12, 2017. [Online]. Available: https://blog.thedataincubator.com/2017/10/ranking-popular-deep-learning-libraries-for-data-science/

[43] L. Linsen. “Point cloud representation,” University of Karlsruhe, Baden-Württemberg, Germany, 2001. [Online]. Available: http://geom.ivd.kit.edu/downloads/pubs/pub-linsen_2001.pdf

[44] R. Amadeo, “Google’s Waymo invests in LIDAR technology, cuts costs by 90 percent,” Ars Technica, Jan. 10, 2017. [Online]. Available: https://arstechnica.com/cars/2017/01/googles-waymo-invests-in-lidar-technology-cuts-costs-by-90-percent

[45] “HDL-64E,” Velodyne. Accessed Nov. 1, 2017. [Online]. Available: http://www.velodynelidar.com/hdl-64e.html

[46] R. Morrison, “Fiducial marker detection and pose estimation from LIDAR range data,” M.S. thesis, Dept. of Comp. Sci., NPS, Monterey, CA, USA, 2010. [Online]. Available: https://calhoun.nps.edu/handle/10945/5411

[47] J. Geng, “Structured-light 3D surface imaging: a tutorial,” *Advances in Optics Photonics*, vol. 3, pp. 128–160, Mar. 2011. [Online]. doi:10.1364/AOP.3.000128

[48] “ZED - depth sensing and camera tracking,” StereoLabs. Accessed Oct. 30, 2017. [Online]. Available: https://www.stereolabs.com/zed/specs/

[49] “HDL-32E,” Velodyne. Accessed Oct. 30, 2017. [Online]. Available: http://velodynelidar.com/hdl-32e.html

[50] “Logitech c920 and c910 fields of view for RGBDtoolkit,” TheRandomLab, Mar. 5, 2013. [Online]. Available: http://therandomlab.blogspot.com/2013/03/logitech-c920-and-c910-fields-of-view.html

[51] “Integration with other libraries,” ROS. Accessed Dec. 4, 2017. [Online]. Available: http://www.ros.org/integration/

[52] “Documentation - Point Cloud Library (PCL),” Pointclouds.org. Accessed Oct. 30, 2017. [Online]. Available: http://pointclouds.org/documentation/tutorials/don_segmentation.php

[53] I. Bogoslavskyi and C. Stachniss, “Fast range image-based segmentation of sparse 3d laser scans for online operation,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7759050/

[54] E. Gouillart and G.Varoquaux, “Image manipulation and processing using Numpy and Scipy,” Scipy-Lectures, Oct. 2017. [Online]. Available: http://www.scipy-lectures.org/advanced/image_processing/

[55] “Drawing functions,” OpenCV. Accessed Dec. 4, 2017. [Online]. Available: https://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California